Generalitat de Catalunya
Centre de Telecomunicacions i Tecnologies de la Informació

Guia de desplegament d' APIs v3.0

Setembre 2026

Índex de Continguts

1 Introducció 3

2 Passos previs al desplegament 3

2.1 Permisos 3

2.2 Instal·lació de programari 6

2.2.1 Git 6

2.2.2 Client VPN 6

2.2.3 API Connect Toolkit 7

2.2.4 Docker 7

2.3 Configuració de l' entorn local 8

2.3.1 VPN 8

2.3.2 API Connect Toolkit i Local Testing Environments (LTE) 10

2.3.3 GitLab 11

2.3.4 GitHub 15

3 Desplegament de APIs 18

3.1 SIC 3.0 18

3.1.1 Consideracions prèvies dins del CTTI 18

3.1.2 Creació de fitxers necessaris 19

3.1.3 Pujada de carpetes i fitxers al repositori 27

3.1.4 Desplegament al Pipeline 30

3.2 SIC+ 41

3.2.1 Consideracions prèvies dins del CTTI 41

3.2.2 Configuració 43

3.2.3 Pujada de carpetes i fitxers al repositori 46

3.2.4 Desplegament utilitzant workflows 48

4 Enllaços d' interès 63

1 Introducció

El present document ha estat elaborat amb la finalitat d' oferir als proveïdors que usen el servei d' API Manager del CTTI d' un conjunt d' instruccions detallades que ajudaran els proveïdors a procedir amb el desplegament d' APIs dels seus projectes.

2 Passos previs al desplegament

Abans de començar amb el desplegament d' APIs, es llisten una sèrie d' accions que s' han de demanar i realitzar per tal de poder dur a terme un desplegament sense dificultats.

2.1 Permisos

Per poder usar les eines i serveis necessaris per al desplegament d'APIs es requereix demanar una sèrie de permisos, que es detallen a continuació:

- GICAR: És l'Identity provider de CTTI. Cal tenir donat d' alta un usuari a GICAR ja que les eines a usar estan integrades amb aquest IDP.

  La gestió de l'alta, baixa, modificació dels usuaris/passwords a GICAR es duu a terme a través del departament de Gestió d'identitats (PMO) dins de CTTI. Cada proveïdor tindrà una persona assignada del PMO, que s'encarregarà de gestionar i redirigir totes les sol·licituds o tràmits a realitzar relacionats amb GICAR.

  Els mecanismes de sol·licitud d'alta a GICAR per a cadascun dels proveïdors es gestionaran de forma independent entre el proveïdor i la persona del PMO assignada a ell. Havent-hi d'aquesta forma mecanismes de sol·licitud diferents per a cada proveïdor.

  ** És recomanat especificar en el correu que es necessita tant credencials GICAR PRE com per a PRO**, per poder accedir tant a l' entorn de Preproducció com el de Producció de GICAR. Com a nota, comentar que per lloar-se als portals d'API Manager, només es fan servir les credencials de PRO de GICAR, tot i que les APIs en PRE sí que poden tenir configurats proveïdors OAuth que usin usuaris de GICAR PRE.

- VPN: Necessària per accedir als recursos interns de CTTI, com ara el GitLab, Jenkins..., per poder realitzar els desenvolupaments per cadascun dels proveïdors.

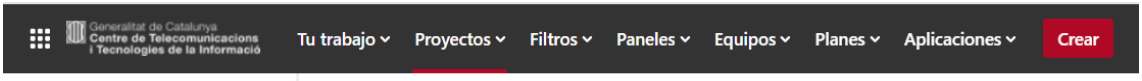
  Cada proveïdor tindrà un conjunt de VPNs assignades que difereixen de la resta de proveïdors.
  La gestió de l'alta, baixa, modificació dels accessos a la VPN es duu a terme igual que l'accés a GICAR, a través de la persona de l'equip de Gestió d'identitats (PMO) dins de CTTI assignada a cada proveïdor, la qual s'encarregarà de gestionar i redirigir totes les sol·licituds o tràmits a realitzar relacionats amb l'accés a les VPNs corresponents.

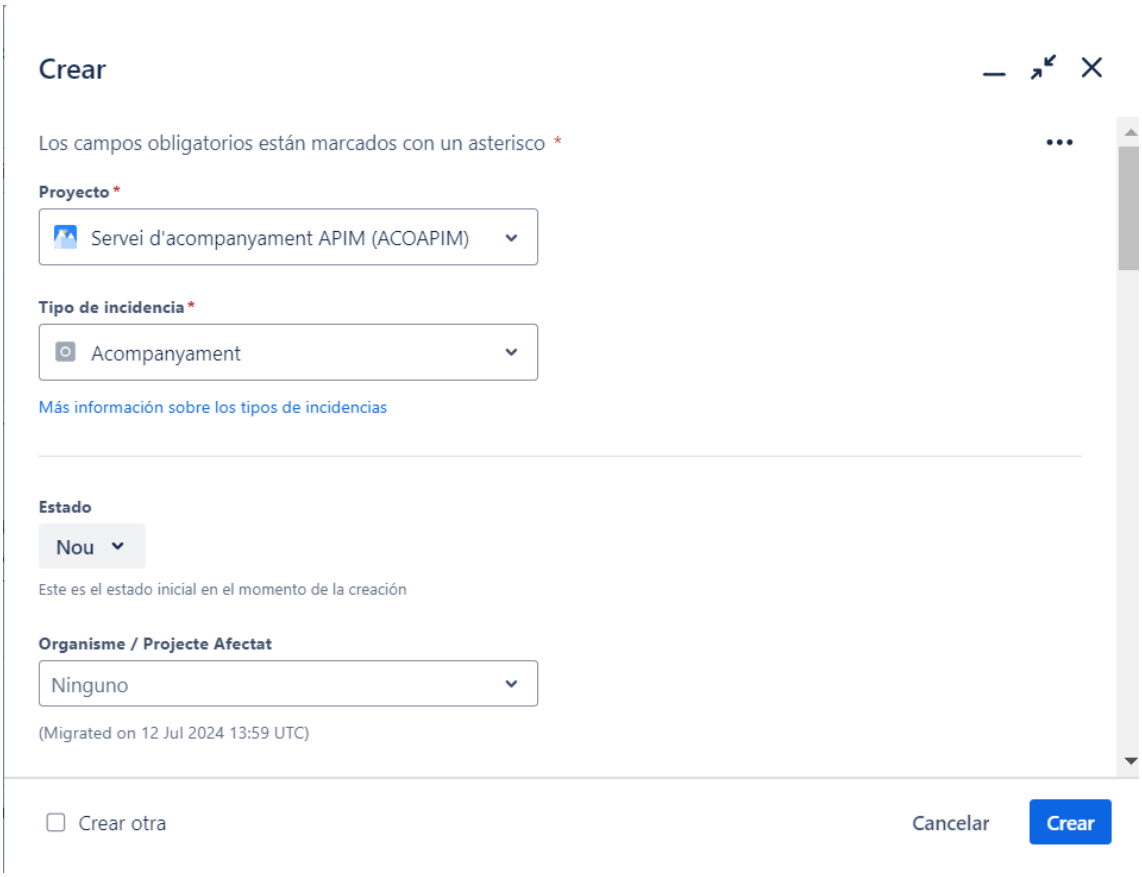
- JIRA: Plataforma que es fa servir per realitzar peticions de suport, incidències, dubtes...

- ** S' haurà de demanar al més aviat possible per poder demanar els permisos restants. **
- L'alta a JIRA es demana manant el portal **Átom** a la secció de **Gestió accés d'usuaris**, sol·licitant accés a peticionar a **ACOAPIM** per poder obrir tiquets d'acompanyament de tot allò relacionat amb API Manager i **ACOCLDSIC** per poder sol·licitar al SIC la configuració necessària per desplegar en **SIC 3.0** o **SIC+**:
    - Gestió accés d'usuaris

El GDI podrà sol·licitar l'alta, la baixa i la modificació d'usuaris en l'aplicació. En el cas que se sol·liciti una petició relacionada amb un projecte JIRA serà necessari adjuntar un correu o document amb la validació del responsable del projecte. Qualsevol dubte que es tingui, es recomana contactar amb l'equip responsable via portal Átom/Remedy o via correu cstd.ctti@gencat.cat.

- **IBM API Connect:** Plataforma per on estan desplegades les APIs i els productes relacionats amb el CTTI.
    - Requereix accés a JIRA**
    - Per sol·licitar el permís a l'IBM API Connect, s'obre un tiquet de JIRA a la següent url: https://cstd-ctti.atlassian.net/jira/software/c/projects/ACOAPIM/issues S'introdueixen les credencials de GICAR i creem una petició nova sol·licitant l'accés corresponent:





A continuació, s'haurien d'emplenar com a mínim els següents camps recomanats:

- **Organisme/Projecte Afectat:** Indicar el nom del projecte/organisme.
- **Si no localitzes el projecte en el desplegable, introdueix-lo aquí**: Indicar el nom del projecte a desplegar, si no es troba en el desplegable.
- **Resum:** Concepte de la petició, per exemple: "*Sol·licitud d'accés a IBM API Connect*".
- **Descripció:** Descripció detallada de la sol·licitud.
- **Codi de servei:** codi del servei per al qual es demana l' acompanyament.

- **GitLab (només si es fa servir SIC 3.0):** És l'eina que es farà servir com a repositori de tots els fitxers relacionats amb el projecte.
  - Un cop donat d' alta el projecte en l' inventari, es crearà al seu torn de forma automàtica el grup de GitLab, assignant el rol de Release Manager a un membre de l' equip qui s' encarrega de crear les carpetes necessàries dins del repositori i de donar permisos a la resta del grup.
  - Per a consultes sobre la gestió de permisos, el funcionament de GitLab..., accedir a la següent url: [Autoservei d'usuaris (gencat.cat).]

- **GitHub (només si es fa servir SIC+):** És l'eina que es farà servir com a repositori de tots els fitxers relacionats amb el projecte.
    - Per consultar com obtenir aquest permís, com fer l'onboarding a GitHub i com configurar tot el necessari en aquesta plataforma, revisar la documentació del SIC+ a Canigó, començant per aquesta url: https://canigo.ctti.gencat.cat/plataformes/ghec/gh-adopcio-model-ghec/

2.2 Instal·lació de programari

2.2.1 Git

Git és un sistema de control de versions distribuït que farem servir en aquest guia per establir connexió i operar a GitLab.

Nota: Cal esmentar que existeixen altres eines de Git disponibles, com Sourcetree, Tortoise Git, Fork..., però en aquest guia es farà servir Git com a exemple.

Els passos per a la seva instal·lació són:

1. Entrar a https://git-scm.com/ i donar-li a la secció Downloads.
2. Escollir la versió que ens correspongui segons el nostre sistema operatiu. En aquest cas, triem 64-bit Git for Windows Setup.

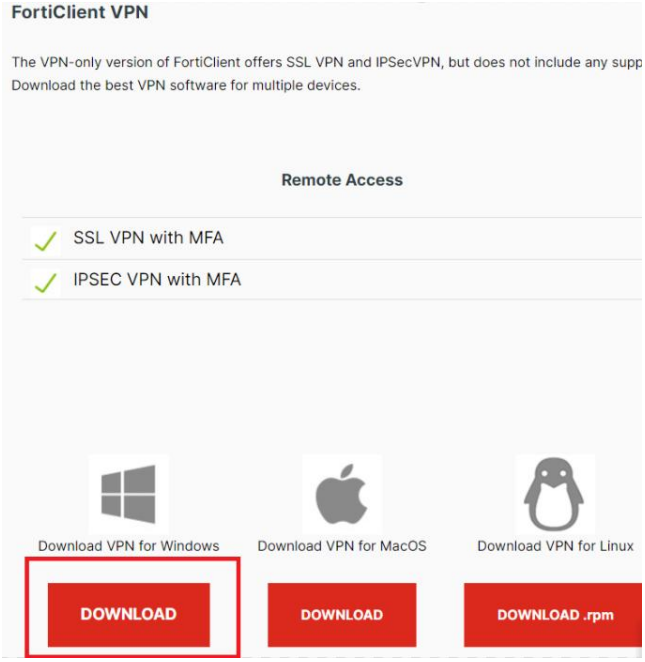
3. Procedim a instal·lar el programari amb totes les configuracions posades en predeterminades.

2.2.2 Client VPN

La VPN és l'eina que farem servir per connectar-nos amb la xarxa interna de CTTI. Per a això, és necessari descarregar-se un client de VPN, com, per exemple, FortiClient VPN, Anyconnect o OpenVPN.

Com a exemple, per descarregar el client de FortiClient VPN, els passos a seguir per a la seva instal·lació serien les següents:

1. Accedir a https://www.fortinet.com/support/product-downloads i descarregar el FortiClient VPN.



2. Després, s'executaria el fitxer "FortiClientVPNOnlineInstaller.exe" per instal·lar el programari. Posteriorment, s' hauria de configurar per poder establir la connexió amb els entorns de CTTI.

2.2.3 API Connect Toolkit

API Connect Toolkit és l'eina facilitada per IBM perquè els desenvolupadors puguin desenvolupar les seves pròpies APIs en el seu entorn local, que compta amb les principals funcionalitats que té l'API Manager. Tota la informació sobre la seva instal·lació i configuració es troba més endavant en el punt 2.3.2.

2.2.4 Docker

Docker és una plataforma de programari que ens permet crear un entorn local de prova per testejar l'API que es desenvolupi. Els passos per a la seva instal·lació són:

1. Accedim a la pàgina oficial de Docker i li donem al botó "Docker Desktop for Windows" per procedir a descarregar-lo.

2. Procedim amb la instal·lació del programari amb totes les configuracions posades en **predeterminades.**

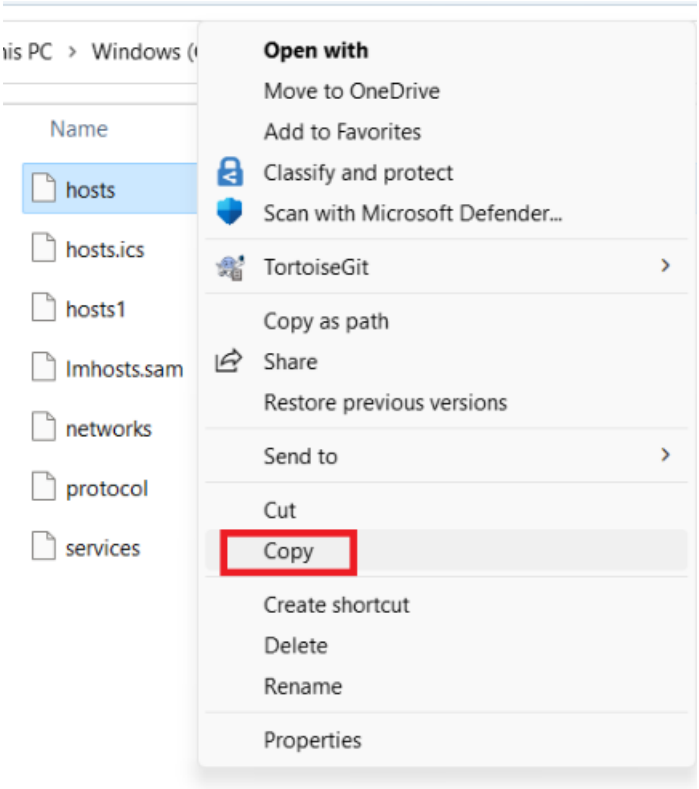
2.3 Configuració de l' entorn local

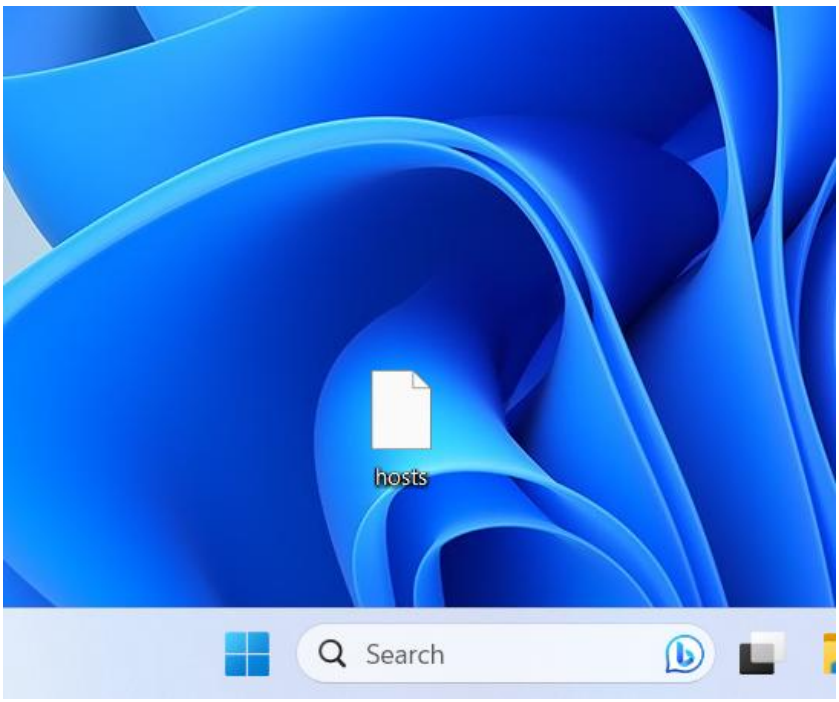
2.3.1 VPN

Tenint en compte que a cada proveïdor se li assigna per part de CTTI un conjunt de VPNs diferents a qualsevol altre proveïdor, el client VPN a usar i la seva configuració haurà de ser indicada i proveïda per la persona del PMO assignada a cada proveïdor.

Com a primer pas, és possible que s'hagi de configurar l'arxiu **hosts**, on es relacionen les adreces IP dels servidors o llocs web amb els seus noms de domini, de manera que puguem accedir als recursos interns de CTTI. Per a això, es llisten els següents passos a seguir per configurar-lo amb les IPs necessàries per al treball en CTTI:

1. Obrim l'explorador d'arxius i ens dirigim a la següent ruta: C:\Windows\System32\drivers\etc
2. Localitzem l'arxiu **hosts** i li donem click dret per realitzar una còpia, la qual enganxem en qualsevol altra ruta (en l'escriptori preferiblement).





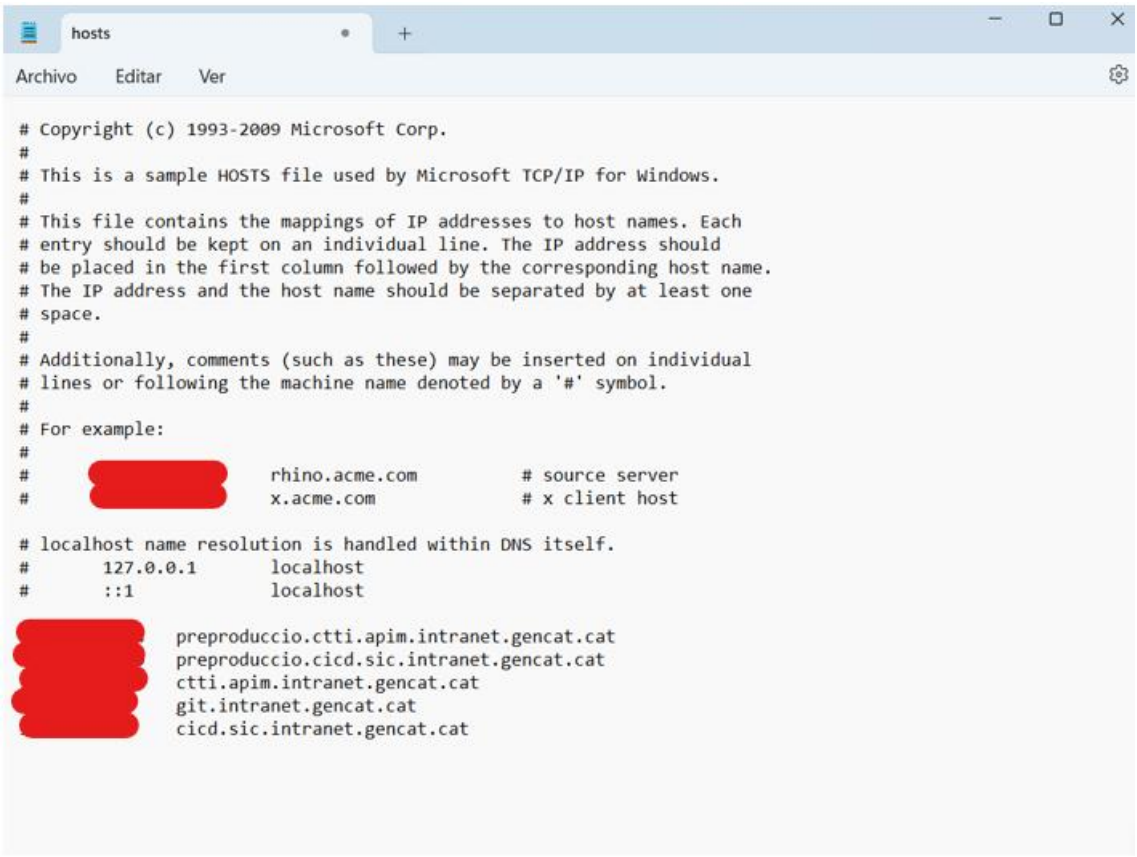
3. Obrim l'arxiu amb un editor de text (bloc de notes) i **hi afegim** les següents IPs dins l'arxiu després de l'última línia comentada (els comentaris dins del fitxer de host es marquen amb #)

```
# localhost name resolution is handled within DNS itself.
#	127.0.0.1 localhost
#	:: 1 localhost
```

Les IPs a afegir (si no es coneixen s' han de demanar als equips responsables) són la següent:

preproduccio.ctti.apim.intranet.gencat.cat (entorn PRIVAT-PRE de l'API-M)*
preproduccio.cicd.sic.intranet.gencat.cat (entorn PRE SIC 3.0)**
ctti.apim.intranet.gencat.cat (entorn PRIVAT de l'API-M)*
git.intranet.gencat.cat (GitLab de CTTI)**
cicd.sic.intranet.gencat.cat (entorn PRO SIC 3.0)**

*Només afegir si es desplegarà en els entorns privats d'API Manager.
**Només afegir si es farà ús de SIC 3.0 per desplegar i no SIC+.



Després d'afegir aquestes rutes IP, li donem a guardar.

4. A continuació, copiarem l'arxiu **hosts** configurat a la ruta: C:\Windows\System32\drivers\etc

2.3.2 API Connect Toolkit i Local Testing Environments (LTE)

Els passos per a la seva instal·lació i configuración de API Connect Toolkit i Local Testing Environments (LTE) es poden trobar en el següent document adjunt (o es pot trobar també a Canigó):

2.3.3 GitLab

GitlịLab és la plataforma de desenvolupament de programari que proporciona eines per a la gestió del cicle de vida d'aplicacions i la col·laboració en projectes de desenvolupament de programari al CTTI en usar SIC 3.0.

2.3.3.1 Requisits previs

Per procedir amb la configuració, cal tenir els permisos demanats següents:

1. Tenir accés a GitLab (https://git.intranet.gencat.cat/). Si no en té, vegeu apartat *2.1*.
2. Tenir accés a GICAR. Si no en té, vegeu apartat *2.1.*
3. Tenir Git instal·lat, vegeu apartat 2.2.1

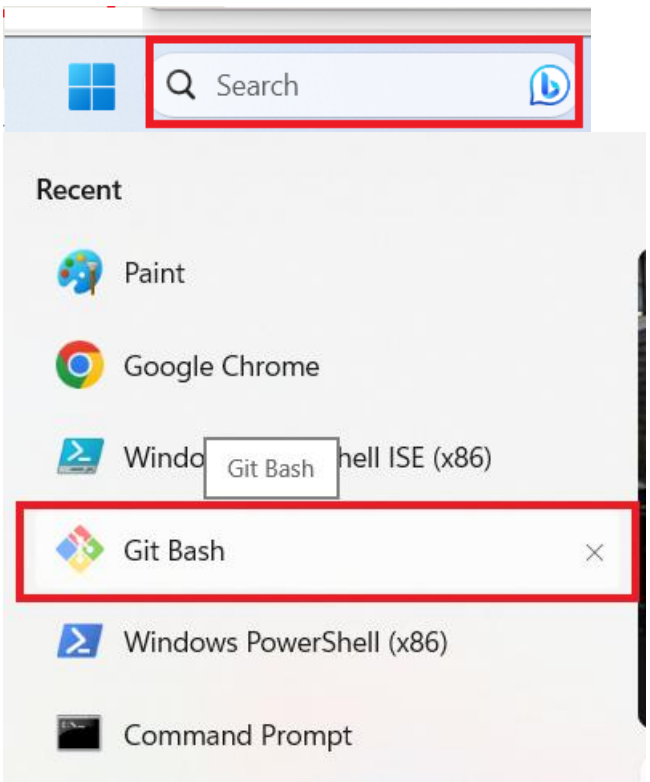
2.3.3.2 Configuració

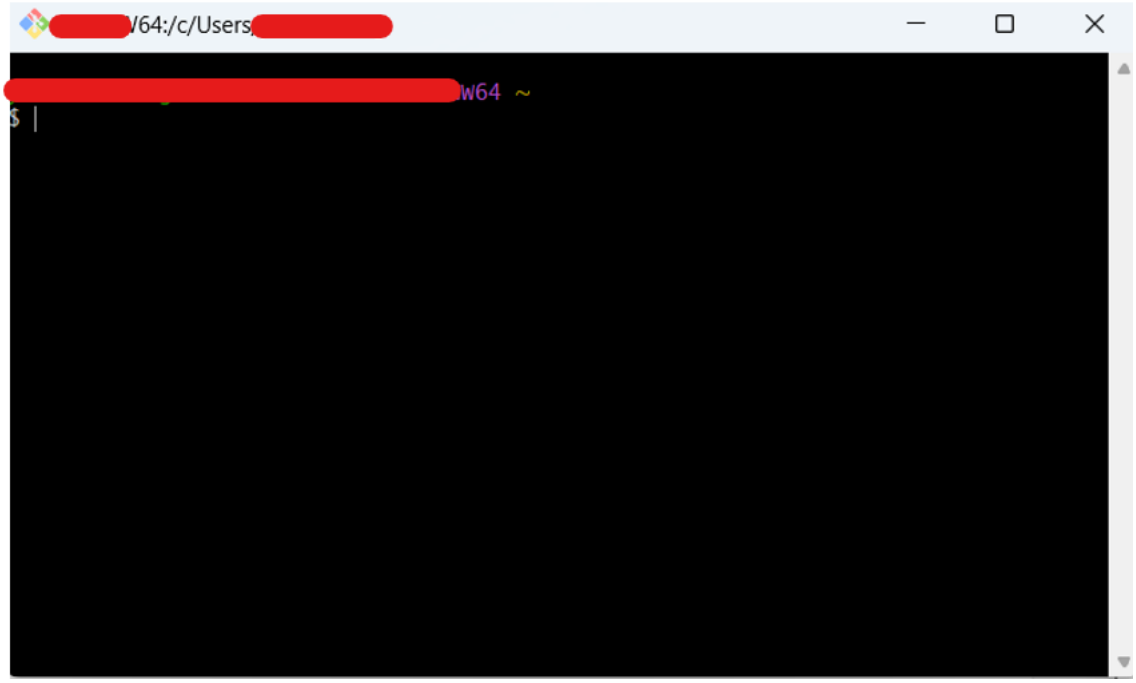
Git és un sistema de control de versions distribuït que farem servir en aquest guia per establir connexió i operar a GitLab.

Nota: Cal esmentar que existeixen altres eines de Git disponibles, com Sourcetree, Tortoise Git, Fork..., però en aquest guia es farà servir Git com a exemple.

La configuració consta dels següents passos a seguir:

1. Obrir el Git Bash que hem instal·lat prèviament, a través de la barra de recerca de Windows.





2. **Creem una carpeta que emmagatzemarà les claus** que utilitzarem posteriorment per connectar-nos amb GitLab.
   `mkdir ~/.ssh/GitLab_ssh`

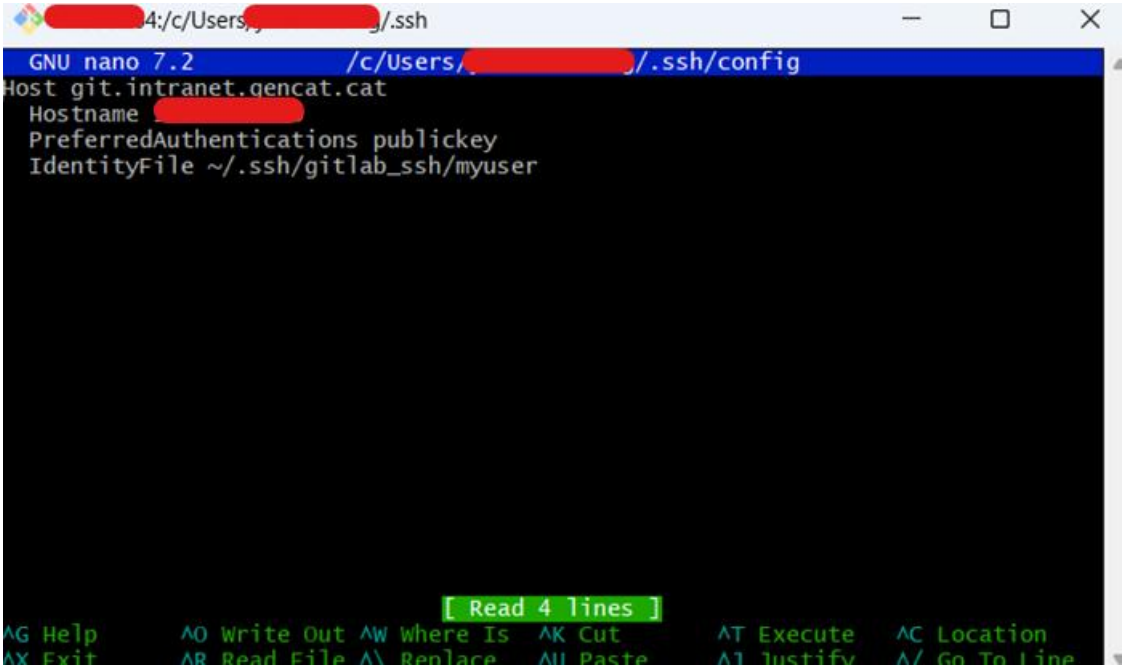
3. Generem aquestes claus mitjançant el següent comandament:
   `ssh-keygen -t rsa -b 4096 -C "<identificador>" -f ~/.ssh/GitLab_ssh/myuser`

   Nota: "<identificador>" se substituirà pel **nom de l'usuari actual**, per exemple:
   `ssh-keygen -t rsa -b 4096 -C alberto.sanchez -f ~/.ssh/GitLab_ssh/myuser`

4. Configurem el fitxer **config** de les claus públiques mitjançant un editor de text anomenat "nano", mitjançant el comandament:
   `nano ~/.ssh/config`

5. Introduïm el següent text dins del fitxer (amb els espais inclosos)

```
Host git.intranet.gencat.cat
  Hostname IP-value-of-gitlab
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/GitLab_ssh/myuser
```



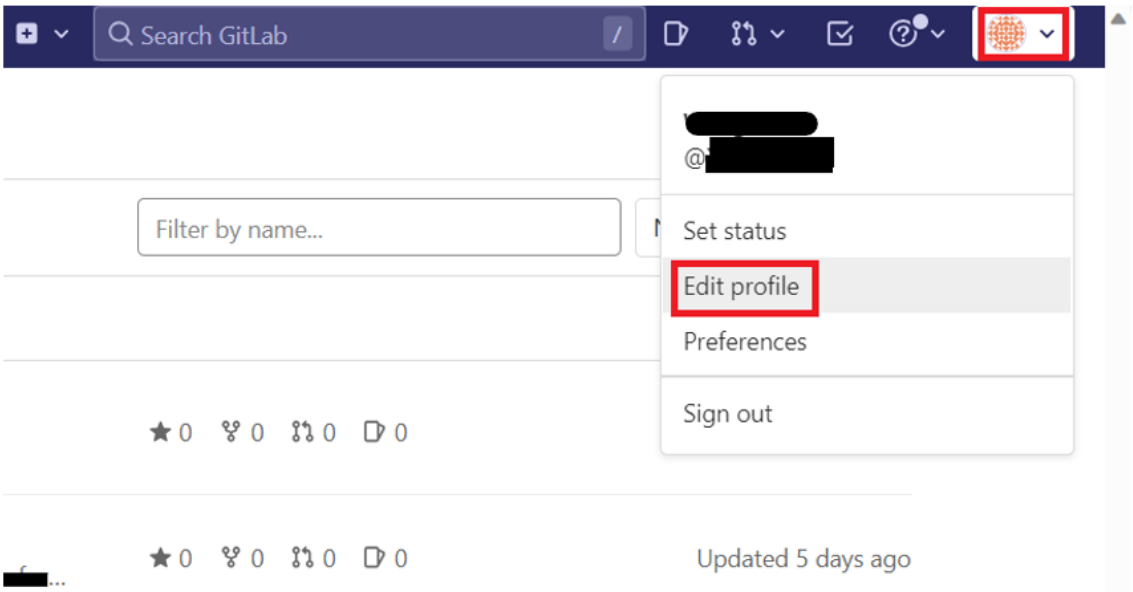
6. Posem les tecles CTRL+O i posem ENTER un cop per guardar els canvis i CTRL+X per sortir de l'editor de text.

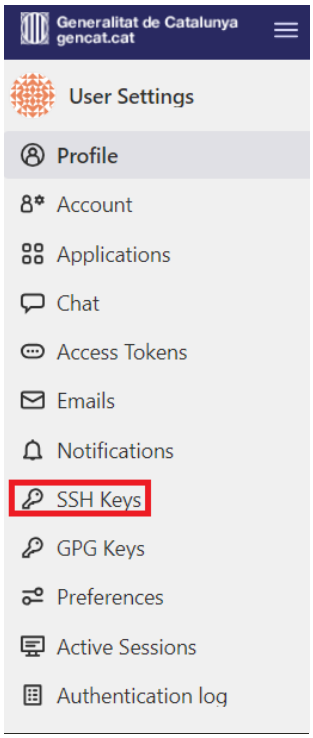
7. A continuació, hem de recuperar la clau pública que hem generat en el pas 3. Per a això, introduïm el següent comandament:
`cat ~/.ssh/GitLab_ssh/myuser.pub`

   Còpiem la clau que es mostra per la pantalla començat per ssh-rsa (l'incloem en la còpia)

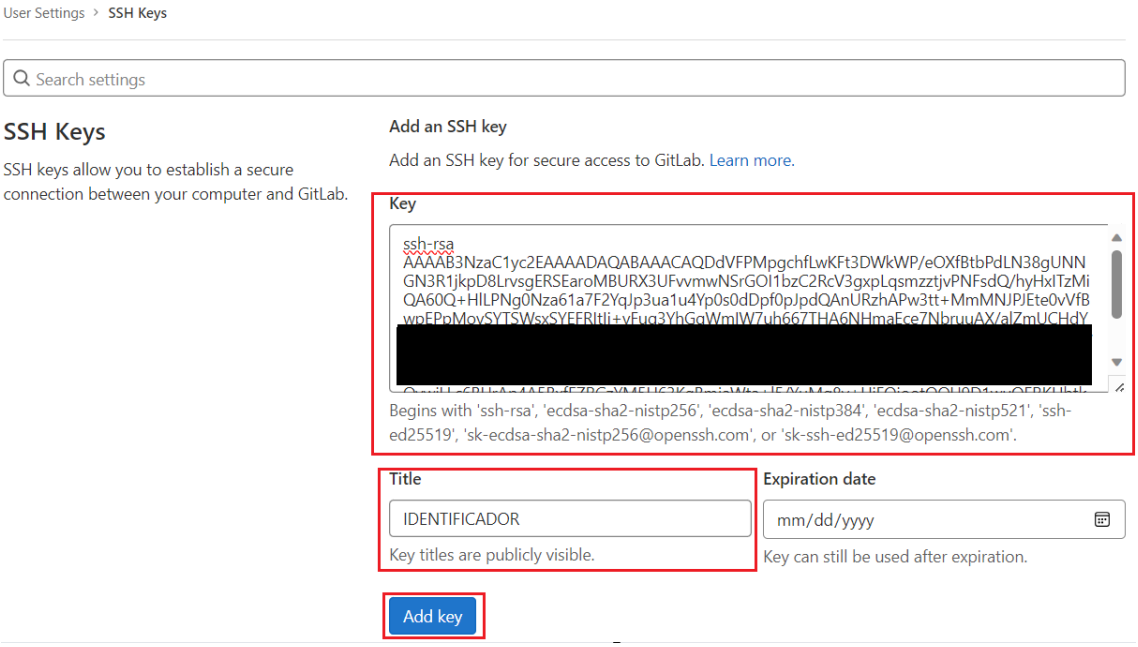
8. Accedim a la pàgina de Git de CTTI mitjançant la següent URL: [Projects · Dashboard · GitLab (gencat.cat)] i ens identifiquem.

9. Importem la clau pública al nostre perfil de GitLab, accedint per "Icona – Edit Profile – SSH Keys".

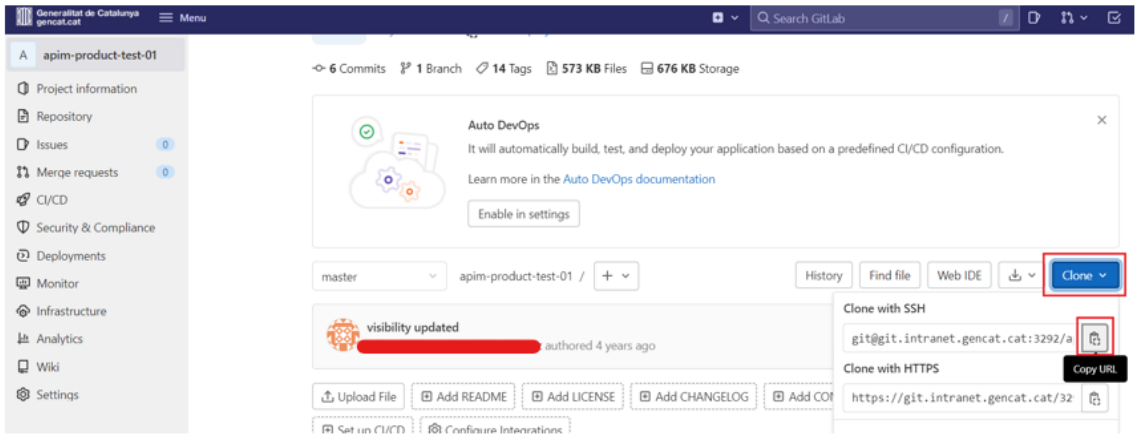




10. Omplim els camps "Key", que correspon a la nostra clau pública, i "Title", que correspon a un identificador que li donem a aquesta clau (Pot ser qualsevol). Posteriorment li donem al botó "Add Key" per finalitzar la importació de la clau.



11. Una vegada finalitzada la configuració de la clau pública, accedeixen al projecte el repositori del qual vulguem clonar i li donem al botó "Clone" seguida pel botó "Copiar" en la secció "Clone with SSH".



12. A continuació, tornem al nostre Git Bash i introduïm el següent comandament per clonar el repositori del nostre projecte.
    `git clone url-repositori`
    Nota: (url-repositori) és la url que hem copiat al pas 11.

    Qualsevol avís que ens pugui sortir per la consola, introduïm la lletra 'i' seguida d'un ENTER.

2.3.4 GitHub

GitHub és la plataforma de SIC+ per la integració d'aplicacions al nou model de CI/CD a cloud públic.

És important comentar que aquest nou model de CI/CD basat en GitHub Enterprise Cloud, d'aquí en endavant GHEC, es recolzarà en el serveis del proveïdor Cloud AZURE per:

- Gestió de grups i usuaris amb Azure Entra ID
- Emagatzematge d'informació (Storage Accounts) i secrets (KeyVaults)

- Ús de Self-hosted Runners

2.3.4.1 Requisits previs

Per procedir amb la configuració, cal tenir els permisos demanats següents:

1. Tenir accés a GitHub (https://ctti.ghe.com/enterprises/ctti/sso). Si no en té, vegeu apartat *2.1*.
2. Tenir accés a GICAR. Si no en té, vegeu apartat *2.1.*
3. Tenir Git instal·lat, vegeu apartat 2.2.1

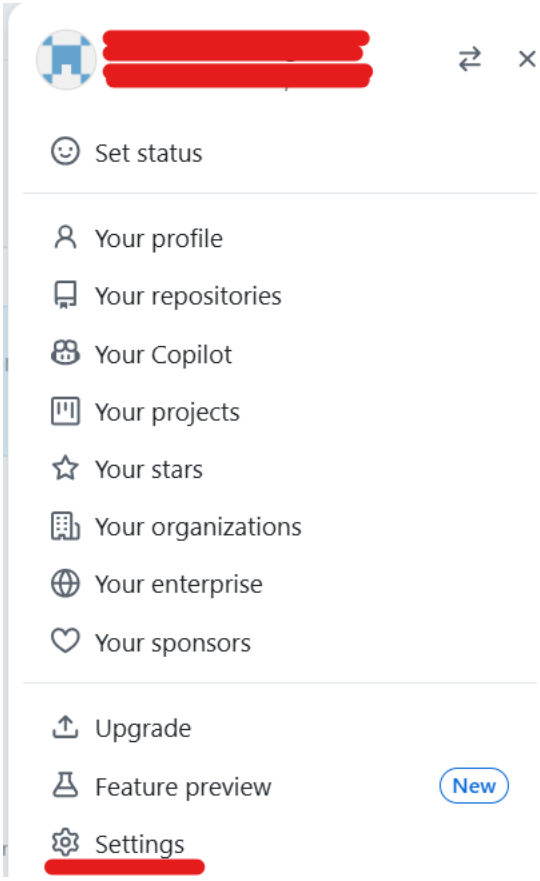
2.3.4.2 Configuració

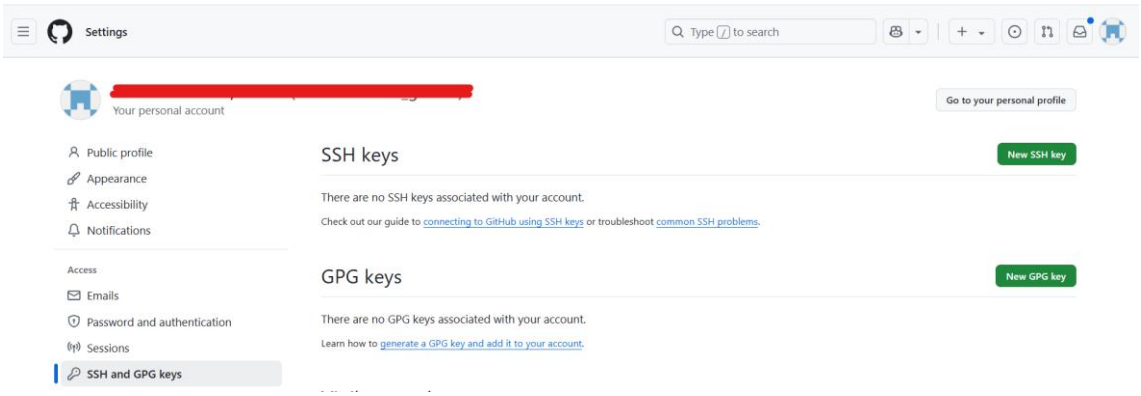
Git és un sistema de control de versions distribuït que farem servir en aquest guia per establir connexió i operar a GitHub.

Nota: Cal esmentar que existeixen altres eines de Git disponibles, com Sourcetree, Tortoise Git, Fork..., però en aquest guia es farà servir Git com a exemple.

La configuració consta dels següents passos a seguir:

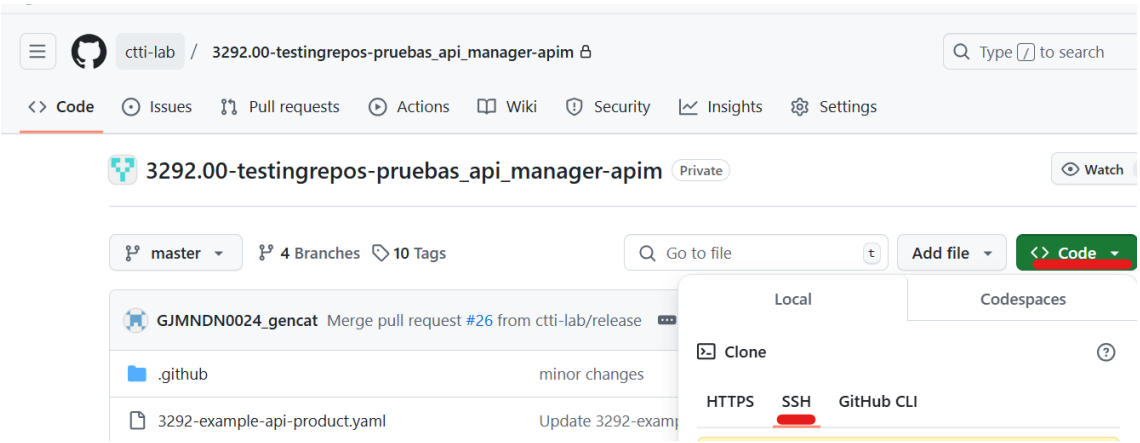
1. Els passos de creació de la clau SSH són similars als que es realitzen en el punt anterior de GitLab, canviant la configuració on correspongui per les dades correctes de GitHub.
2. Luego, accedim a la pàgina de GitHub de CTTI mitjançant la següent URL: https://ctti.ghe.com/enterprises/ctti/sso), i ens identifiquem amb GICAR.
3. Importem la clau pública al nostre perfil de GitHub, accedint per "Icona – Settings - SSH and GPG Keys".





4. Posem en "New SSH Key". Omplim els camps "Key", que correspon a la nostra clau pública, "Key type" i "Title", que correspon a un identificador que li donem a aquesta clau (Pot ser qualsevol). Posteriorment li donem al botó "Add SSH Key" per finalitzar la importació de la clau.

5. Una vegada finalitzada la configuració de la clau pública, accedeixen al projecte el repositori del qual vulguem clonar i li donem al botó "Code" seguida pel botó "SSH" i, per últim, "Copy url to clipboard".



6. A continuació, tornem al nostre Git Bash i introduïm el següent comandament per clonar el repositori del nostre projecte.

   ```
   git clone url-repositori
   ```

   Nota: (url-repositori) és la url que hem copiat al pas 11.

   Qualsevol avís que ens pugui sortir per la consola, introduïm la lletra 'i' seguida d'un ENTER.

3 Desplegament de APIs

A continuació, es detallen els passos a realitzar per desplegar les APIs, tant usant el pipeline de Jenkins de SIC 3.0 com usant els workflows de GitHub de SIC+.

3.1 SIC 3.0

3.1.1 Consideracions prèvies dins del CTTI

L' organització de catàlegs, espais i productes és la següent:

- Actualment CTTI compta amb quatre catàlegs segons el tipus d' entorn i el tipus de visibilitat:
    - ***privat_pre*** i ***privat*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a la **Intranet**.
    - ***public_pre*** i ***public*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a l' **Internet**.
- Cada codi de diàleg disposarà d'un espai propi amb la nomenclatura "CD" + <codi_diàleg>. Per exemple: "CD0192".
- Un producte és una agrupació d'APIs i plans que les acompanyen (unitat mínima a versionar, desplegar i subscriure).

Un cop fet el desenvolupament amb API Designer, caldrà exportar els YAMLs de definició del producte i els seus APIs per dipositar-los en el sistema de gestió de codi font (SCM - Source Code Management) del SIC d'acord amb les següents premisses:

- Cada producte es correspondrà amb un projecte dins del codi de diàleg adequat, de manera que tota la gestió posterior de pipelins i creació de peticions Remedy/JIRA s'eixirà a l'aplicació corresponent. Per aquest mateix motiu, no està contemplat la creació de subgrups de projectes, encara que l' eina ho permeti.
- Els projectes poden tenir tantes branques com siguin necessàries, però sempre s' haurà d' incloure la branca MASTER, ja que el contingut d' aquesta branca serà amb el que treballaran els pipelins de desplegament per defecte. No obstant això, el sistema permetrà opcionalment desenvolupar el codi font associat a la branca EVOLUTIUS.
- Aquest repositori no és un entorn de desenvolupament, per la qual cosa només les persones assignades com Release Managers seran les encarregades de consolidar el codi i lliurar-lo. Aquest codi font ja haurà d' estar validat en entorns de desenvolupament i es lliurarà quan es decideixi distribuir en els entorns dels serveis TIC centrals.
- Els pipelins seran les encarregades de generar els TAGS de Release de codi corresponents quan es desplegui amb èxit a Producció.

3.1.2 Creació de fitxers necessaris

De cara al funcionament correcte del pipeline cal que s' hagin creat prèviament uns arxius que s' allotjaran al GitLab, bé al repositori del projecte o al projecte del SIC. Aquests fitxers són usats pel pipeline tant per a l' aprovisionament i creació del pipeline, com perquè el pipeline reculli el detall de la configuració de l' API a desplegar. Els fitxers són:

3.1.2.1 ACA (Arxiu generat pel desenvolupador)

Es tracta d' un **arxiu de configuració d' aplicacions imprescindible per a** qualsevol projecte **que s' ha d' allotjar en el repositori del projecte**, la funció del qual dins de l' autoservei de pipelins del CTTI, és la de configurar l' aplicació de tal manera que tingui activat el mode **Autoservei de Pipelins**, que és un servei mitjançant el qual es poden generar **automàticament** pipelins de construcció i desplegament d'aplicacions, amb el treball

col·laboratiu dels **proveïdors d'aplicacions i d'infraestructures** i sense la intervenció de l'equip del SIC. Per a més informació, consultar [Autoservei de pipelines (gencat.cat)].

Per tal de configurar la integració en el SIC, tots els projectes hauran de disposar d' una carpeta en el primer nivell i, dins d' aquesta carpeta, caldrà crear el fitxer ***aca.yml***. Exemple de ruta :/sic/aca.yml

Un exemple del fitxer ACA seria el següent:

```yaml
version: 2.0.0 # aca schema version
info:
  version: 1.0.1
  description: Api Connect Product
global-env:
  - APIC_PRODUCT_FILE: 'product.yml'
  - APP_NAME: 'technical-product'
  - APIC_PRESERVE_ASSEMBLY: true
components:
 - deployment:
      environments:
        - name: privat_pre
          actions:
            deploy:
              steps:
                - execution:
                    env:
                      - APIC_TARGET_URL: 'https://backend/pre'
                      - APIC_GATEWAY_SERVICES: 'azure-intranet-pre'
        - name: privat
          actions:
            deploy:
              steps:
                - execution:
                    env:
                      - APIC_TARGET_URL: 'https://backend/pro'
                      - APIC_GATEWAY_SERVICES: 'intranet-pro,aws-intranet-pro'
notifications:
  email:
    recipients:
      - recipiente@correo.es
```

On els punts clau per tenir en compte serien els següents:

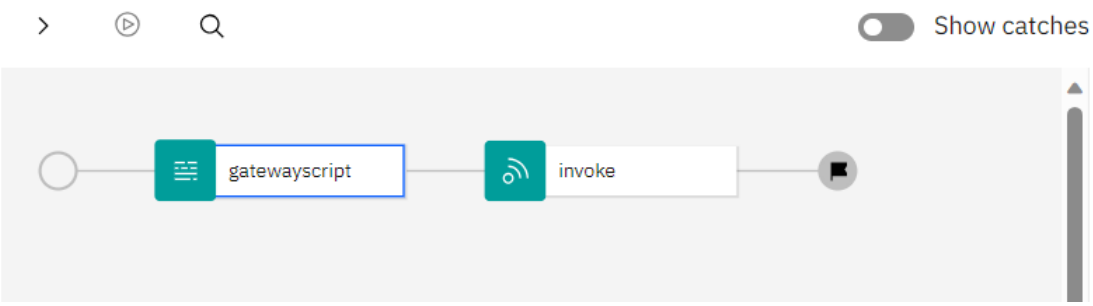
- **Version:** versió (independent de la versió de l'aplicació) que s'utilitza per fer seguiment de l'arxiu de configuració.
- **Environments:** Entorns als quals es desplega l' aplicació, així com el seu ordre i la modalitat de desenvolupament desitjada. Actualment CTTI compta amb quatre catàlegs segons el tipus d' entorn i el tipus de visibilitat:
    - ***privat_ pre*** i ***privat*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a la **Intranet**.
    - ***public_pre*** i ***public*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a l' **Internet**.

  En l'exemple exposat a dalt el desplegament de l'API es realitzaria en els entorns de privat_pre (preproducció) i privat (producció) pertanyents al catàleg d'intranet.

Nota: S'hauria d'indicar en aquest fitxer només els entorns on es vagi a desplegar el producte i l'API (podent ser un o diversos, en funció de la necessitat del projecte).

A més, cal destacar que en el cas que el pipeline no permet desplegaments mixtos; és a dir, perquè es pugui desplegar a l'entorn de Producció Externa "public", és necessari haver desplegat explícitament a l'entorn de Preproducció Extern "public_pre".

- Notifications: Definició d'adreces de correu electrònic on notificar accions manuals en espera i resultats de l' execució.
- APP_NAME: Nom corresponent al producte que es vagi a desplegar. Aquest nom no ha de contenir el codi de diàleg del projecte, ja que aquest serà inclòs de manera automàtica després de realitzar-se el desplegament amb el format < codi >-<APP_NAME>. Per exemple, si el nom del camp del YAML del producte té configurat el valor 3292-technicalproduct, el valor que s'indicarà en APP_NAME del fitxer ACA seria technicalproduct.
  En cas que no es configuri aquesta variable en el ACA, se li assignarà per defecte el nom del projecte pel nom del repositori de GIT.
- APIC_PRODUCT_FILE (opcional): Ruta i nom del fitxer descriptor per al desplegament de l'aplicació a l'API Manager. La variable només serà requerida en cas que la ruta i/o nom del fitxer difereixi del suggerit.
  - *Valor per defecte: product.yml*
- APIC_PRESERVE_ASSEMBLY (opcional): Curricular que indica si s'ignora o no la regla que reemplaça l'acoblament, a través dels valors *true* o *false*, conservant l'acoblat original del YAML de l'API (secció assembly al YAML) creat pel desenvolupador, en cas que el camp s'informi amb el valor *true*. Això permetrà desplegar l'API amb lògica personalitzada.
  En el cas que no s'inclogui aquesta variable o que s'informi a *false*, s 'executaran les regles establertes en el pipeline, que substitueixen la lògica de l'API a desplegar l'acoblament, per una configuració estàndard en la qual només es realitza una trucada a l'endpoint definit en el fitxer aca.yaml (*APIC_TARGET_URL* o *APIC_TARGET_URL_{N}*). L' acoblat resultant seria el següent:

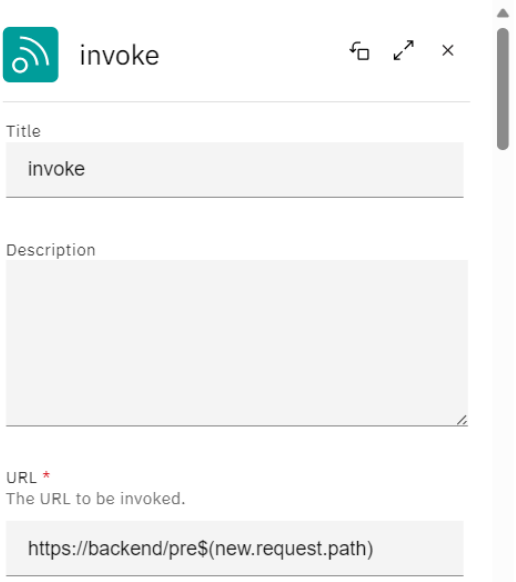


- Gatewayscript. *Contindria la següent informació:*

```
1  const requestPath = context.get('request
       .path');
2  const newRequestPath = requestPath.replace
       (/^\/\d*\.*\d*/, "");
3  context.set('new.request.path',
       newRequestPath);
4
```

Es recull la via d'accés *(request.path)* de l'API i es guarda en una constant. Se li dona format a aquesta constant i es genera la variable de context '*new.request.path*', que posteriorment serà utilitzada en la política d'Invoke.

- Invoke. *Contindria la següent informació: <APIC_TARGET_URL>$(new.request.path)*



- APIC_TARGET_URL: URL de destinació de les APIs si és comú a totes, tot i que pot conviure amb *APIC_TARGET_URL_{N}* per especificitats.
  NOTA: si el valor de APIC_PRESERVE_ASSEMBLY és true, s' ignorarà el valor que es posi en aquest camp.
  - *Exemple: APIC_TARGET_URL: 'https://backend/api'.*

- APIC_TARGET_URL_{N} (opcional): URL de destinació definida per a una API específica dins del producte, que no coincideixi amb la URL de destinació comuna per a totes. Permetent d' aquesta manera definir especificitats, encara que pot conviure amb *APIC_TARGET_URL global*.
  NOTA: si el valor de APIC_PRESERVE_ASSEMBLY és true, s' ignorarà el valor que es posi en aquest camp.
  - Format de la clau*: APIC_TARGET_URL_{0-*9a-*zA-Z}*
  - Format del valor: *<api-file-name-with-extension>:<target-url>*Per consultar més informació sobre el fitxer **Arxiu de Configuració d'Aplicacions**, consultar el punt d'annex.
  - *Exemple: APIC_TARGET_URL_1: 'api_1.0.0.yml:https://backend/apio'*

- APIC_GATEWAY_SERVICES (opcional): Llista dels gateways de l'espai destí on s'ha de desplegar el Producte, expressada com a cadena separada per comes. Si s'omet, val 'All' o és una cadena buida (''), el desplegament es fa a tots els gateways associats a l'espai del catàleg destí, mantenint la retrocompatibilitat amb el comportament anterior que hi havia. Si es declara una llista no buida, el desplegament es restringeix exclusivament als gateways indicats. Si algun gateway de la llista no existeix a l'espai, el pipeline falla controladament abans d'iniciar cap publish, sense desplegament parcial.
    - Valors possibles de gateways:
        - Privat PRE: azure-intranet-pre, aws-intranet-pre, intranet-pre
        - Privat PRO: azure-intranet-pro, aws-intranet-pro, intranet-pro
        - Public PRE: azure-internet-pre, aws-internet-pre, internet-pre
        - Public PRO: azure-internet-pro, aws-internet-pro, internet-pro
    - Exemple: *APIC_GATEWAY_SERVICES: 'azure-internet-pre,aws-internet-pre'*

Seguint aquest format, podríem construir el nostre fitxer ACA, per poder desplegar el nostre producte amb els seus corresponents APIs, en els entorns requerits dins de CTTI.

3.1.2.2 ACS (Arxiu generat per SIC)

L' ACS és l' arxiu de configuració generat i proporcionat per part de l' equip de SIC, el qual conté la configuració interna per al funcionament del pipeline i altres configuracions necessàries.

Un exemple d' un fitxer ACS és el que es mostra a continuació:

```yaml
version: 2.0.0
global-env:
  - CONTAINER_REGISTRY_PROJECT: project
  - MAVEN_OPTS: '-Dmaven.deploy.skip=false'
  - JOB_TEMPLATE: 'job-volume.yaml'

  - APIC_ORGANIZATION: 'org'
  - APIC_SPACE: 'space'
  - APIC_PLANFILE: 'file.yaml'
  - APIC_CATALOG_DES: 'des'
  - APIC_CATALOG_PUB: 'null'

components:
  - deployment:
      actions:
        deploy:
          steps:
            - container:
                image:
                  remote:
                    name: registreimatges.sic.intranet.gencat.cat/gencat-sic/apic-deployer:3.0
                resources:
                  limits:
                    cpu: 2000m
                    memory: 2Gi
                  requests:
                    cpu: 500m
                    memory: 512Mi
                execution:
                  commands:
                    - run.sh $APIC_ORGANIZATION $APIC_SPACE $APIC_PLANFILE $APIC_CATALOG $APIC_PRODUCT_FILE $APIC_PLAN_MAP
      environments:
        - name: privat
          confirmation:
            required: true
          staging: true
          production: false
          actions:
            deploy:
              steps:
                - execution:
                    credentials:
                      - id: harbor-credentials-id
                        basic:
                          username: REGISTRY_USERNAME
                          password: REGISTRY_PASSWORD
                    env:
                      - NAMESPACE: null
                      - APIC_CATALOG: null
                      - JOB_ENVS: DB_TYPE=null|DB_HOST=null|DB_PORT=null|DB_DATABASE=null|DB_USERNAME=null|DB_PASSWORD=null
```

```yaml
        - name: privat_pre
          confirmation:
            required: false
          staging: true
          production: false
          actions:
            deploy:
              steps:
                - execution:
                    credentials:
                      - id: harbor-credentials-id
                        basic:
                          username: REGISTRY_USERNAME
                          password: REGISTRY_PASSWORD
                    env:
                      - NAMESPACE: null
                      - APIC_CATALOG: null
                      - JOB_ENVS: DB_TYPE=null|DB_HOST=null|DB_PORT=null|DB_DATABASE
                        =null|DB_USERNAME=null|DB_PASSWORD=null
```

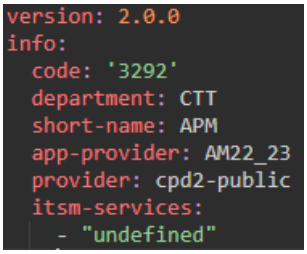
Alguns dels camps més importants de l' ACS són:

- **Remot**: Imatge i versió del contenidor constructor a utilitzar entre les disponibles en el registre d'imatges corporatiu (per a més informació consultar [Registre d'imatges (gencat.cat)]( ).
- **Resources**: recursos a assignar per a la correcta construcció del projecte (CPU i memòria).
- **Global-env:** Relació de variables globals necessàries per a l' execució del pipeline.
- **Credentials:** Credencial que s' aporta a la imatge perquè es pugui executar.
- **Commands:** Operacions (scripts) que s'executen per a la construcció del projecte.

3.1.2.3 ACD (Arxiu generat per SIC)

L'ACD és **l'arxiu de configuració de departament,** el contingut del qual detalla la informació sobre l'àrea tècnica del servei del CTTI, Datacenter a usar on es desplegarà l'API, l'identificador del proveïdor/projecte i el codi del departament o àrea tècnica del CTTI. **L' equip de SIC és el responsable de proporcionar i mantenir aquest arxiu.**

Un exemple d' un fitxer ACD és el següent:



Alguns dels camps més importants de l' ACD són:

- **Versió**: versió de l' arxiu de configuració.
- **Code**: Codi identificatiu del departament.
- **Department**: Nom del departament.
- **App-provider:** Identificador del proveïdor de l' aplicació.

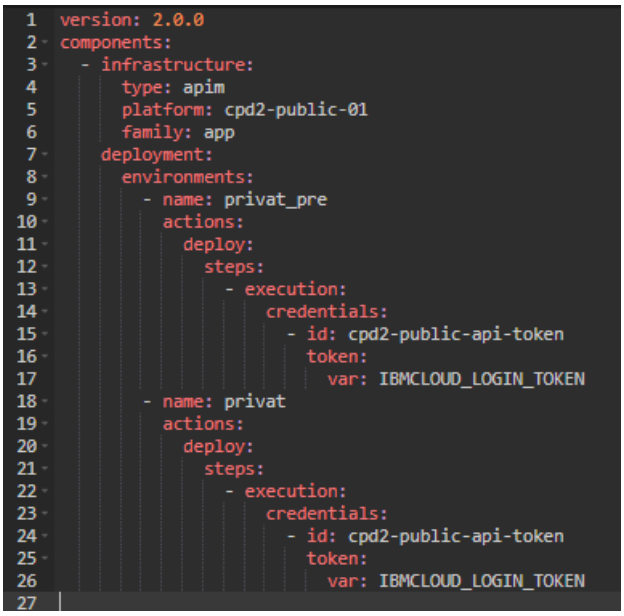
3.1.2.4 ACI (Arxiu generat per SIC)

L'ACI és l **'arxiu de configuració d'infraestructures** (pot tenir-ne més d'un per aplicació-projecte), el nom del qual (sense incloure l'extensió) serà l'identificador facilitat al proveïdor d'aplicacions. Serà la responsabilitat del **proveïdor d' infraestructures** mantenir actualitzada aquesta informació i de notificar al proveïdor d' aplicacions quan s' hagi realitzat algun canvi sobre el fitxer.

Es tracta d' un arxiu de text en format YAML, que serà responsabilitat del proveïdor d' infraestructures mantenir-lo actualitzat, en el qual s' ha d' aportar la següent informació:

- **Versió**: versió de l' arxiu de configuració.
- **Components**: secció que recull tots els recursos de la part d' infraestructures. Actualment, només existeix el detall de cada infraestructura.
- **Infrastructure**: detall de les infraestructures incloses en aquest arxiu de configuració.

Un exemple d' un fitxer ACI és el següent:



3.1.2.5 map.yaml (arxiu opcional per a l'operativa REPLACE)

Es tracta d'un fitxer opcional que s'allotja al repositori del projecte, a la mateixa altura que els fitxers de productes i APIs (no en cap subcarpeta). El seu nom ha de ser exactament map.yaml. Si no existeix, s'aplica el comportament per defecte.
Aquest fitxer defineix el mapeig de plans entre el Producte sortint (l'antic, que se substitueix) i el Producte entrant (el nou) en l'operativa REPLACE. Segueix el format estàndard documentat per IBM per al mapping file del comandament apic products:replace:

```
plans:
  - source: <plan_sortint_1>
    target: <plan_entrant_1>
  - source: <plan_sortint_2>
    target: <plan_entrant_2>
```

Quan cal aportar-lo:

- Si els plans del Producte entrant tenen el mateix nom que els del sortint i el mapeig per defecte (source == target) cobreix tots els plans, el fitxer no és necessari.
- Si hi ha renombrats de plans, fusions (N source → 1 target), afegit de plans nous o qualsevol divergència de plans entre Producte sortint i entrant, el fitxer és obligatori; en cas contrari, el REPLACE fallarà controladament.

Comportament del pipeline:

- Sense map.yaml i plans coincidents → REPLACE OK; les subscripcions migren del plan source al plan target corresponent (source == target).
- Sense map.yaml i plans no coincidents → el pipeline falla controladament amb un missatge clar.
- Amb map.yaml correcte → REPLACE OK; les subscripcions actives migren del plan source al plan target indicat.

- Amb map.yaml incomplet o referenciant un target inexistent al Producte entrant → el pipeline falla controladament amb un missatge clar.

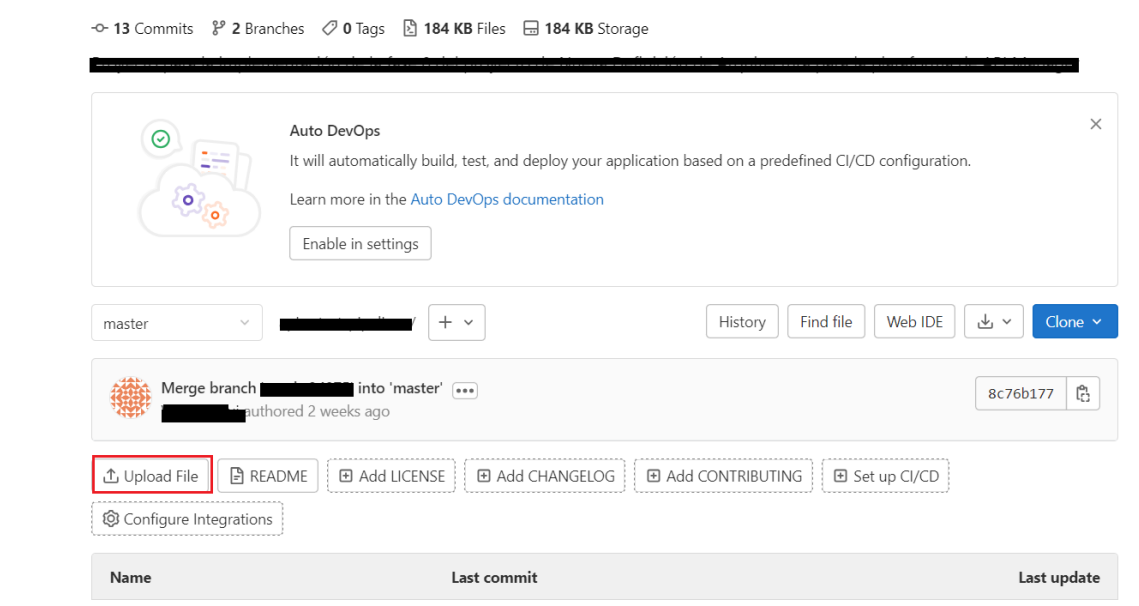
Exemple:

```
plans:
  - source: default-plan
    target: bronze
  - source: silver
    target: silver
  - source: gold
    target: new-plan
```

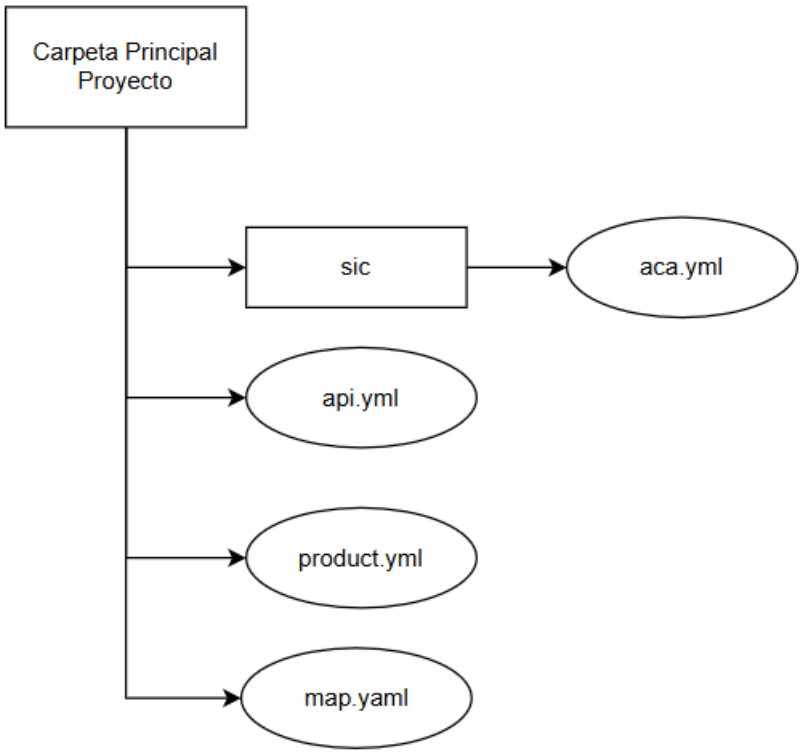
3.1.3 Pujada de carpetes i fitxers al repositori

Una vegada finalitzades les fases de disseny tant de les APIs com dels productes, on s' han configurat els fitxers YAML corresponents a cadascun d' ells, s' hauran de pujar aquests fitxers al repositori assignat al projecte, per posteriorment procedir amb el desplegament entre entorns.
A continuació, es mostren els passos a seguir en el cas de l'ús de GitLab des de la interfície web. En cas d' usar un client Git com Fork, TortoiseGit o Sourcetree, l' execució dels passos serà diferent, encara que el concepte darrere de cada pas és el mateix:

1. Accedim al GitLab (gencat.cat) i ens lloguem amb la credencial de GICAR.
2. Accedim al repositori del projecte que vulguem desplegar i li donem al botó "Upload File".



3. Triem els fitxers que vulguem pujar (ho pugem un a un), tenint en compte que l'estructura del repositori ha de ser el següent:

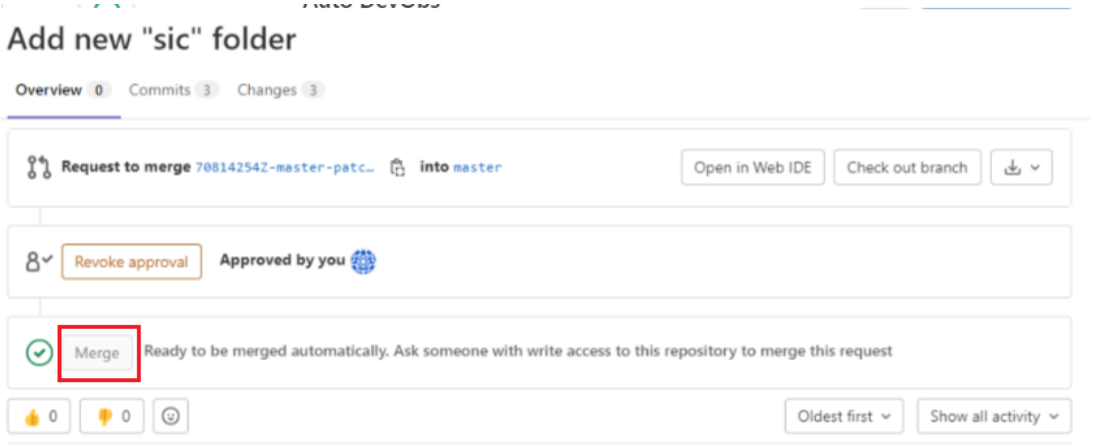


Nota:

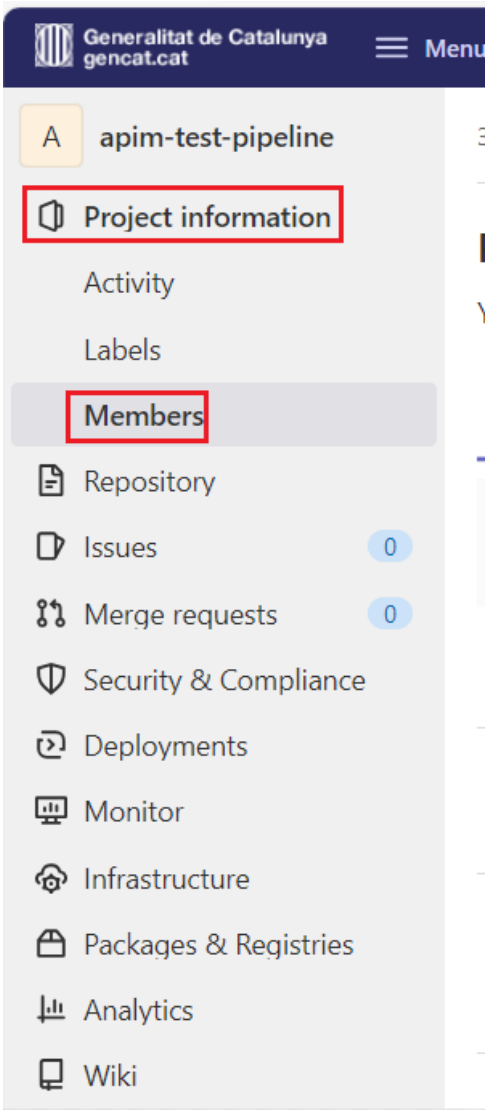
- *Un producte pot contenir diverses APIs, per la qual cosa un projecte de GitLab pot contenir la quantitat d'APIs que es necessiti, sempre que estiguin referenciats en el corresponent producte (camp "apis" del YAML del producte).*
- *En el cas que es puguen APIs de tipus SOAP o SOAP-TO-REST, cal pujar l'arxiu ZIP de WSDL a la mateixa altura que el fitxer YAML de l'API. El YAML s'haurà d'haver generat abans amb l'ajuda de l'API Designer/Toolkit, que permet importar un fitxer WSDL i generar un YAML amb una API SOAP o SOAP-TO-REST definida en base a l'esmentat WSDL.*
  Un exemple d'un fragment del iml d'una api generada en base al WSDL seria el següent:

```yaml
wsdl-definition:
  wsdl: wsdlFileName.zip
  service: wsdlNameService
  port: wsdlNameImplPort
  soap-version: '1.0'
```

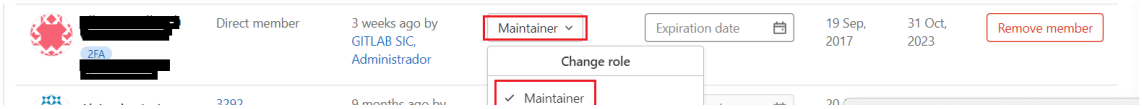
4. En el cas que s'hagi creat una branca nova diferent al "master" (branca principal), el següent pas és demanar el mergeig de la branca creada amb la branca principal. Per a això, es necessita l'aprovació d'un membre que tingui el permís de Maintainer (concedit pel Release Manager) o Release Manager (concedit pel SIC).



5. Per donar permís de *Maintainer* com *Release Manager*, naveguem per la secció "Project Information – Members".



6. Seleccionem el membre al qual vulguem assignar el permís de *Maintaner*.

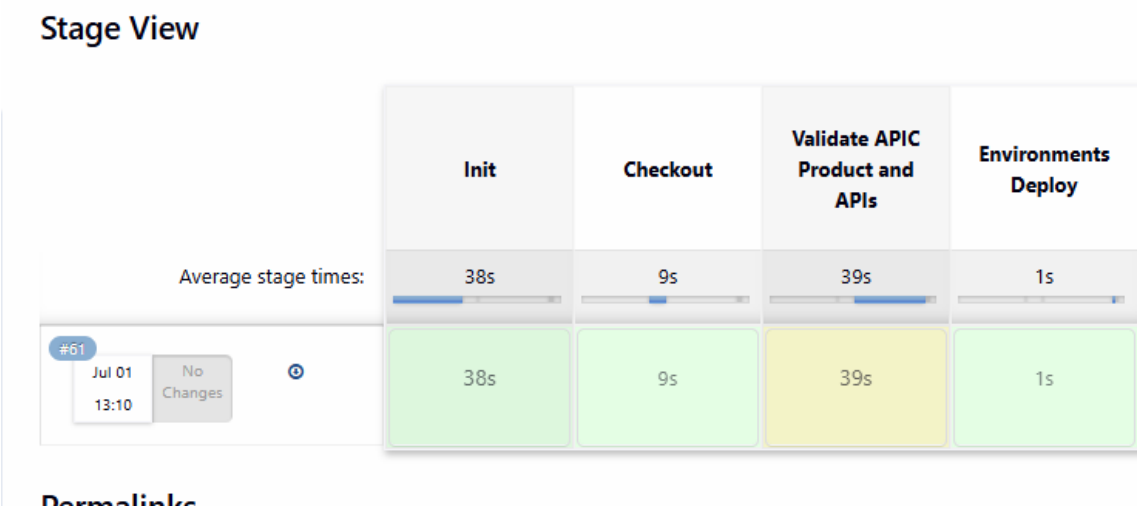


3.1.4 Desplegament al Pipeline

Un cop tinguem preparat el repositori amb els fitxers necessaris al GitLab, procedirem a realitzar les proves de pipeline a l'entorn de preproducció per provar la funcionabilitat de l'API desenvolupada (alternativament es pot provar a l'entorn local amb Toolkit, vegeu apartat 2.3.2).

3.1.4.1 Validació corporativa de governança al pipeline

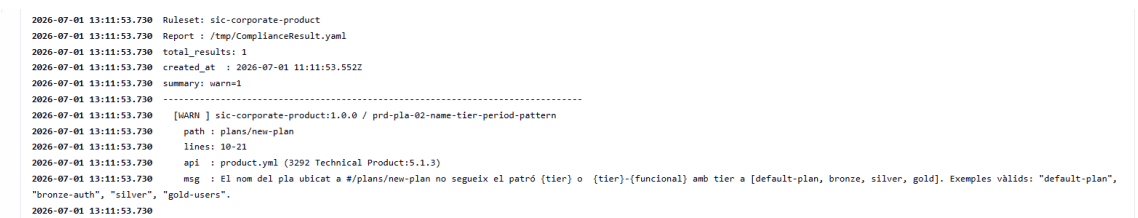
Com a part del pipeline de desplegament de SIC 3.0, s'executa de forma automàtica una validació de governança contra els rulesets corporatius sic-corporate-api (regles sobre el document OpenAPI) i sic-corporate-product (regles sobre el descriptor de Producte).



Aquesta validació actua com a compuerta de qualitat prèvia al desplegament:

- Si la validació detecta findings de severitat error, el pipeline bloqueja l'execució abans de cap publish.
- Si només hi ha findings de severitat warn, l'execució continua però queda registre dels avisos als logs.

El detall de cada finding (regla, path, missatge) i el resum agregat queden disponibles als logs del job i a l'artefacte JSON adjuntat a l'execució. La descripció funcional completa de les regles és al Disseny tècnic i a l'ERQ corporatius dels rulesets sic-corporate-api / sic-corporate-product.



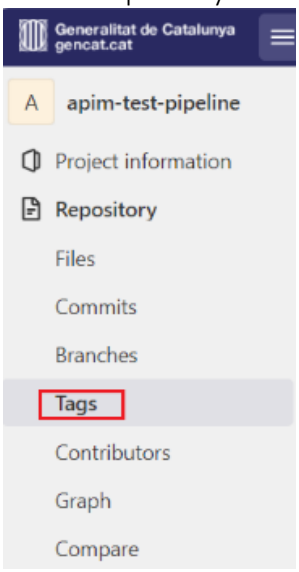
3.1.4.2 Passos per executar el pipeline

Un cop pujats els arxius necessaris, es procedirà amb el desplegament del producte i l' API a l' entorn de preproducció, el procediment del qual és el següent:

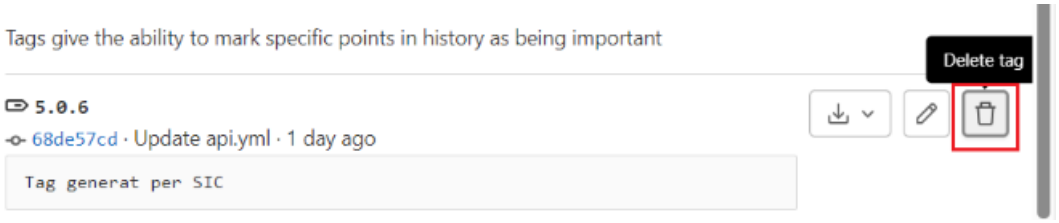
1. Comprovació i eliminació de l' existència del tag associat a la versió que volem desplegar.

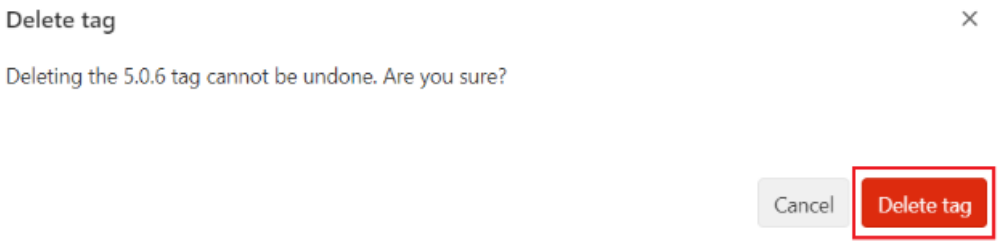
   Nota: *Cada vegada que es desplega un producte a totes les entorns configurats a l'aca.yml, es genera un tag associat a la versió a desplegar, perquè des del SIC es pugui tenir una traçabilitat de les versions desplegades. Per tant,* ***a partir del primer desplegament, si es vol desplegar la mateixa versió del producte****, per exemple, perquè s' hagi solucionat algun problema detectat en els desplegaments després de fer unes proves durant el cicle de desenvolupament, s' ha d' aplicar el workaround establert,* ***esborrant de forma manual el tag creat associat a la versió del producte al GIT, per part del Release Manager o Maintainers assignats al projecte****.*
   *(En el cas que no s'apliqui aquest workaround, es produirà l'error de "****Ja hi ha un tag definitiu amb la versió: X.x.x****" pel duplicat del tag al GIT).*

   - Accedim a la pestanya de tags a la part esquerra del Git



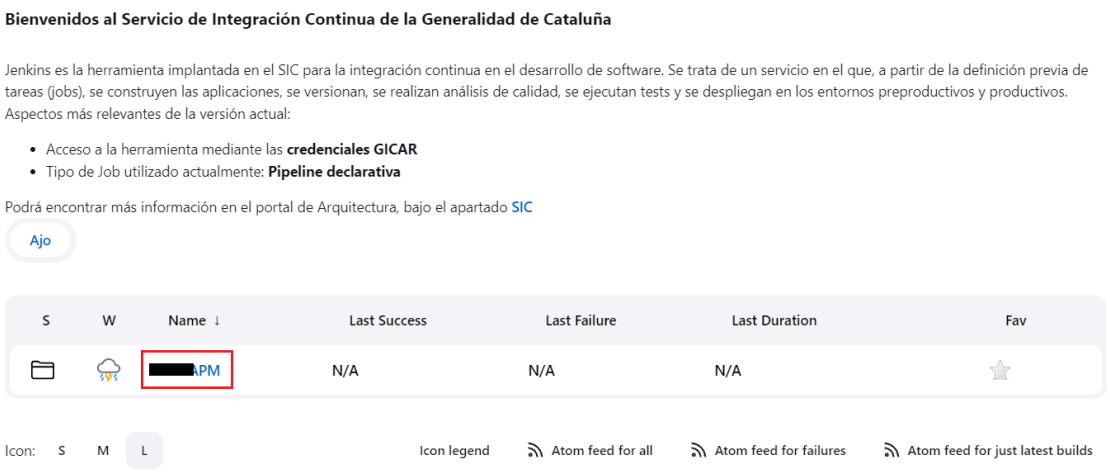
   - Seleccionem la versió a esborrar i procedim amb la seva eliminació.



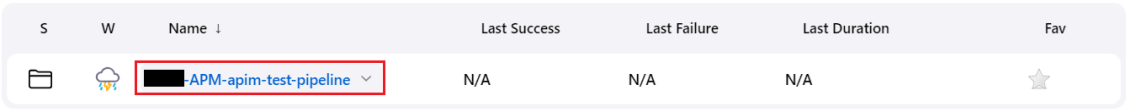


- Després d'haver-nos cerciorat de l'eliminació del tag corresponent a la versió que volem desplegar, podem procedir amb el següent pas.

2. Accedim a Jenkins Dashboard [Jenkins] (gencat.cat) i ens identifiquem amb les credencials de GICAR.
3. Dins del Dashboard, accedeixen al projecte que correspon:

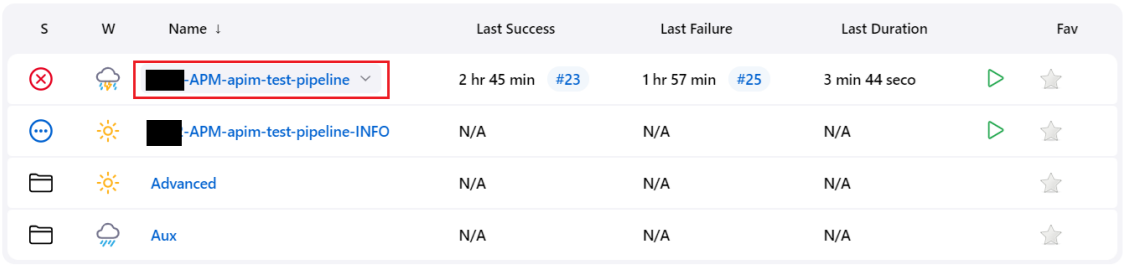


4. Seleccionem la carpeta del pipeline creat per part del SIC.

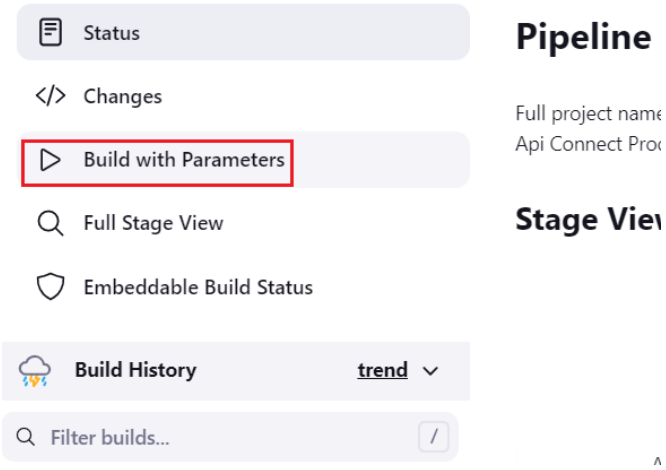


5. ... i el pipeline que vulguem provar.

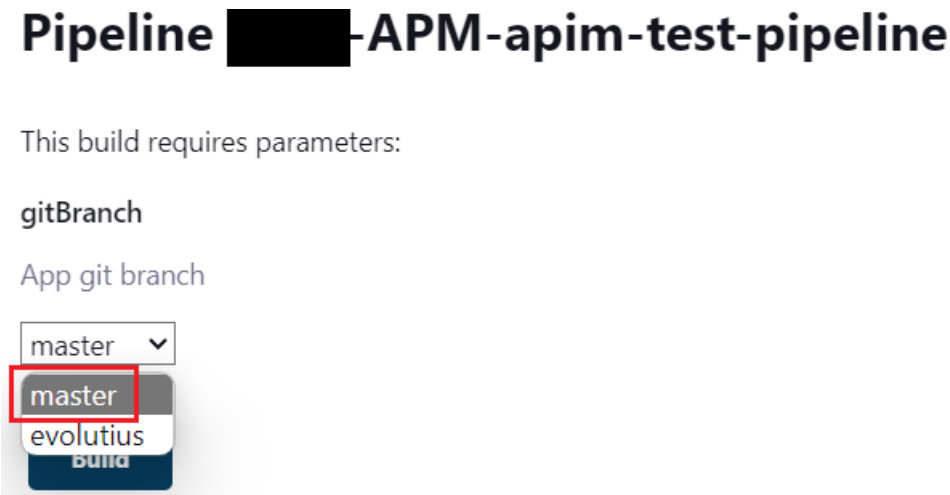
Nota: *Es poden executar diversos tipus de pipeline, com ara PUBLISH, per publicar un producte i api, INFO per obtenir informacions dels productes desplegats en un catàleg concret, i altres pipelins com DEPRECATE, RETIRE, SUPERSEDE, DELETE i REPLACE que es troben a la carpeta "Advanced". Per informar-se del funcionament de cadascun dels pipelins, consultar el següent enllaç:* Autoservei de pipelines (gencat.cat).

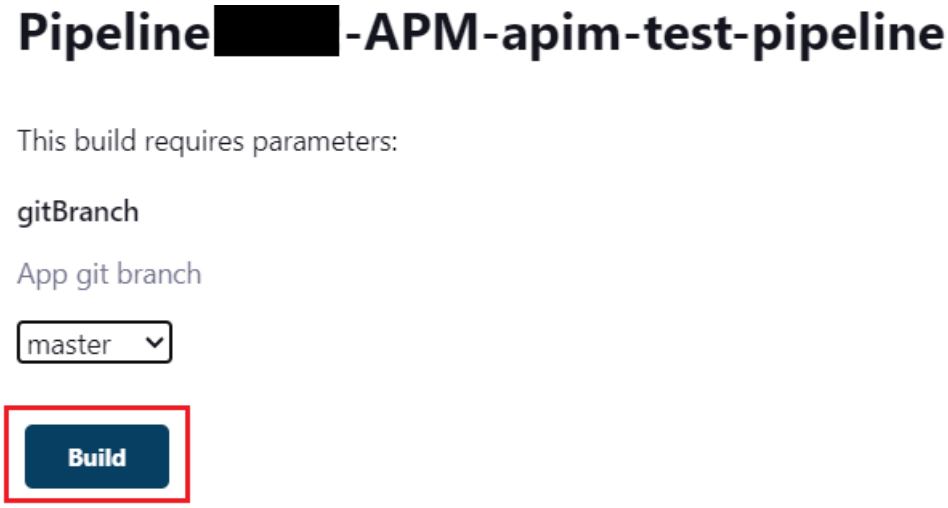


6. Al menú de l'esquerra, seleccionem "Build with Parameters".

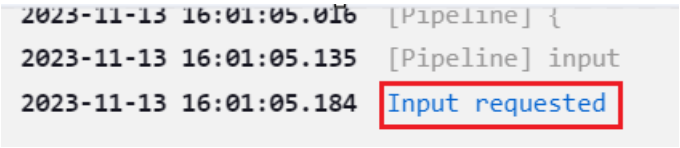


7. Seleccionem la branca "master" per desplegar la nostra branca principal del repositori i executem el pipeline donant-li a "Build".

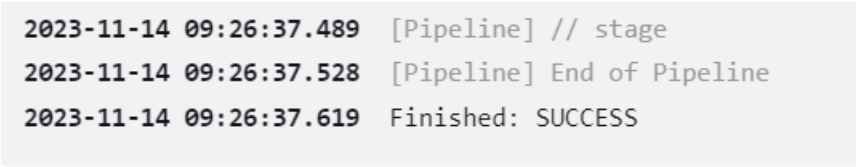




8. Al llarg de l' execució, per la consola del pipeline es demanarà a l' agent perquè introdueixi els paràmetres necessaris per procedir amb l' execució. Li donem click sobre l'enllaç proporcionat quan correspongui.

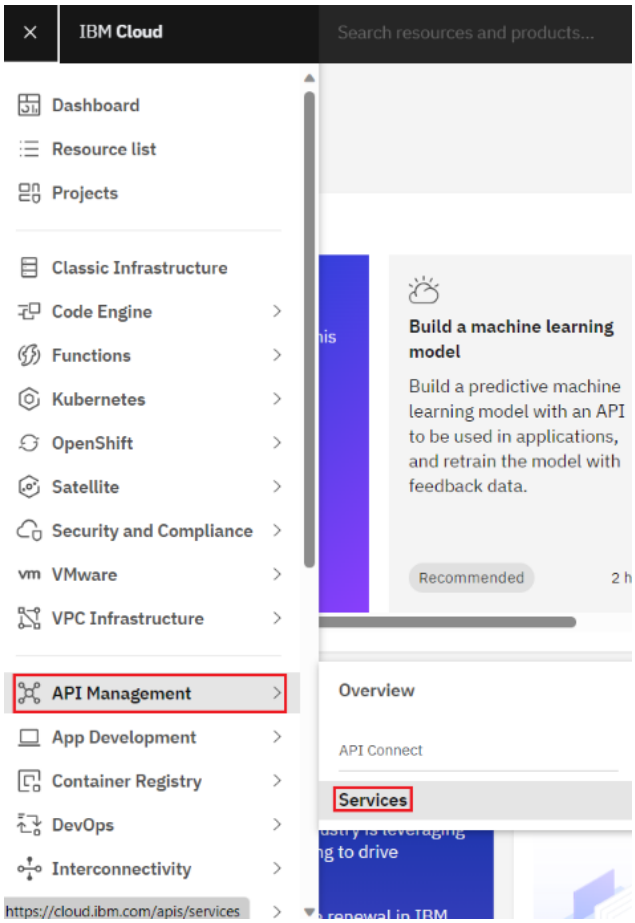


9. En el cas que el pipeline s' ha executat correctament, es mostrarà per la consola el missatge de SUCCESS.



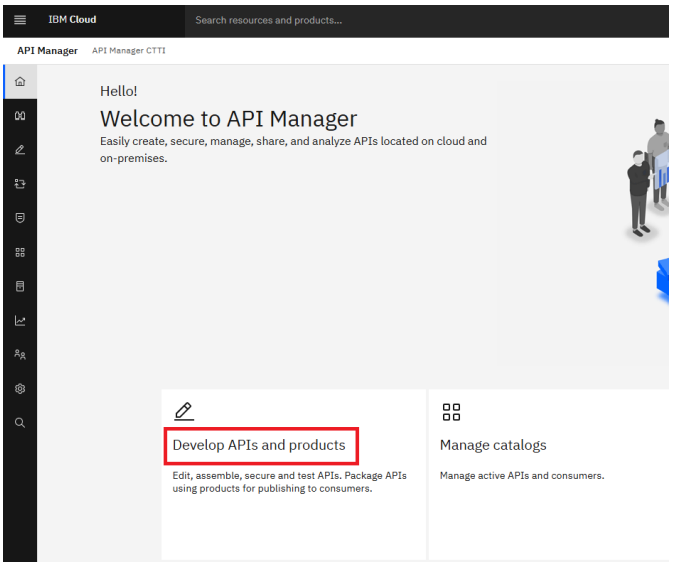
10. Després, per comprovar que el producte i l'api s'ha desplegat correctament, ens dirigim a https://cloud.ibm.com/authorize/gicar i ens identifiquem amb les credencials de GICAR.

11. Un cop dins, accedim al menú ubicat a l'esquerra i accedim a "API Management – Services".

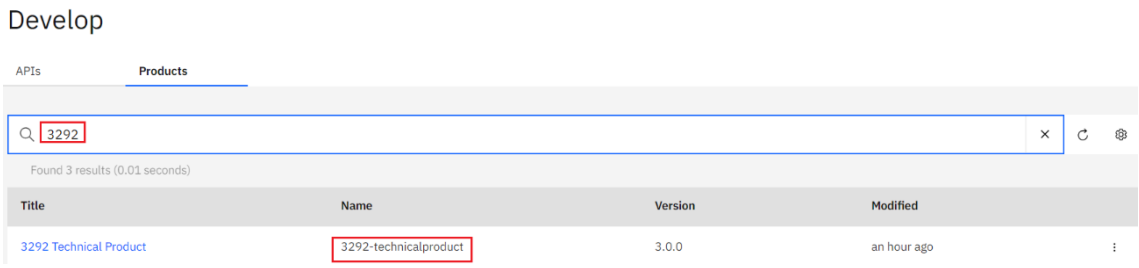


12. Ara accedeixen dins de CTTI.

13. Accedim a la secció "Develop APIs and products".

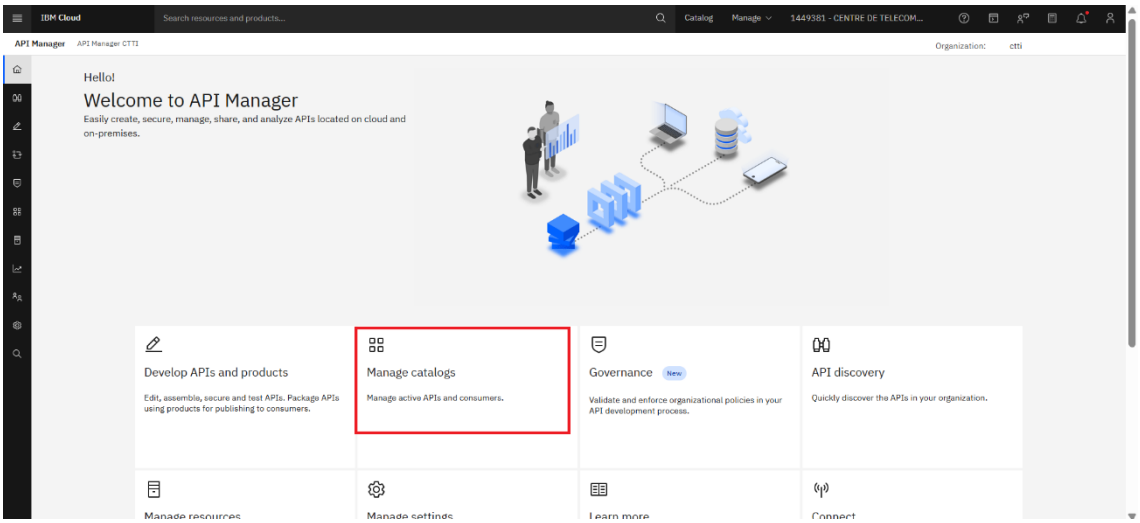


14. A la barra de recerca de productes, introduïm el nom del nostre producte desplegat i confirmem que s'ha desplegat correctament el draft del producte.

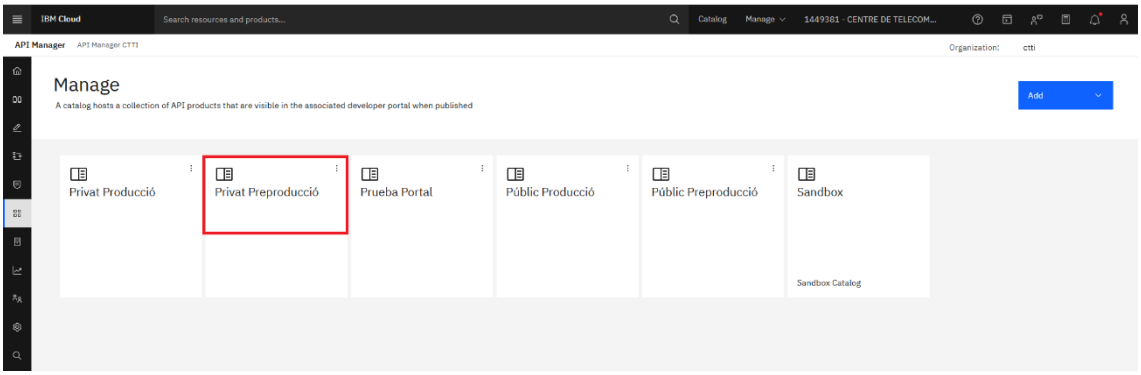


15. A la barra de recerca d'APIs, introduïm el nom de la nostra API desplegada i confirmem que s'ha desplegat correctament el draft de l'API.

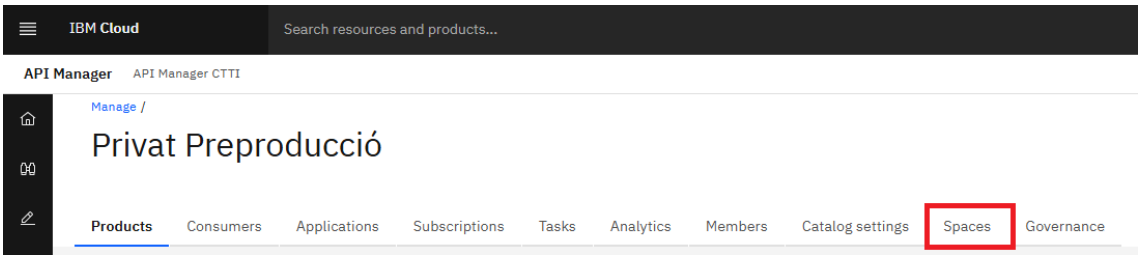
16. A continuació, comprovem que el producte s'ha desplegat al catàleg. Per a això, accedim a la secció "Manage Catalogs".

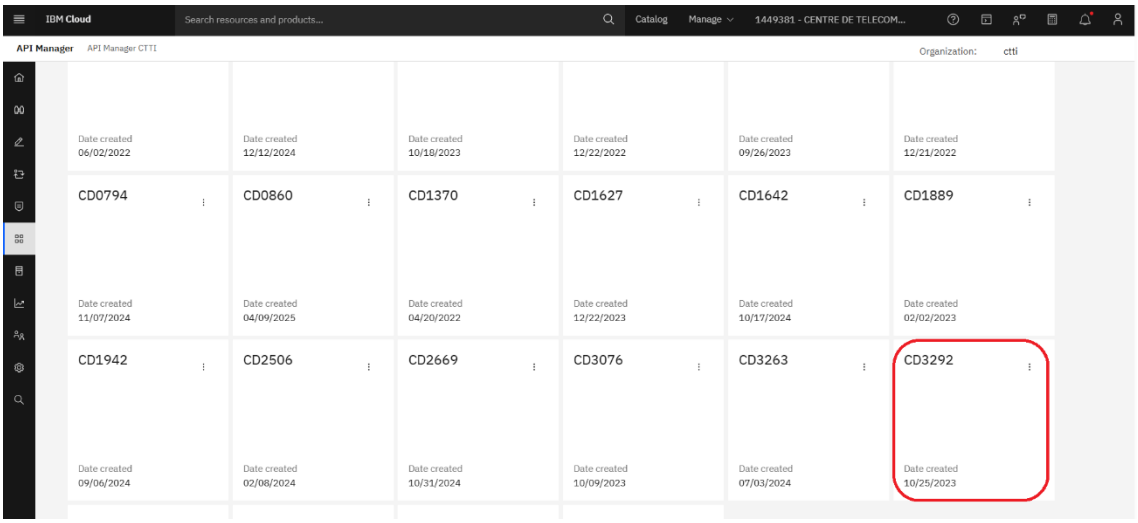


17. Accedim al catàleg per on s'ha desplegat el nostre producte, que en aquest cas és Privat Preproducció.

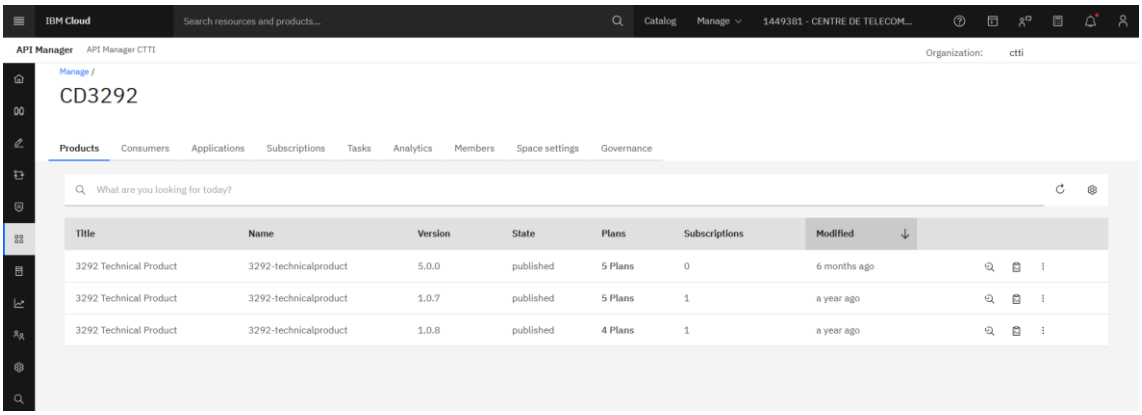


18. Ara entrem al nostre espai assignat dins del catàleg, que és el CD3292.



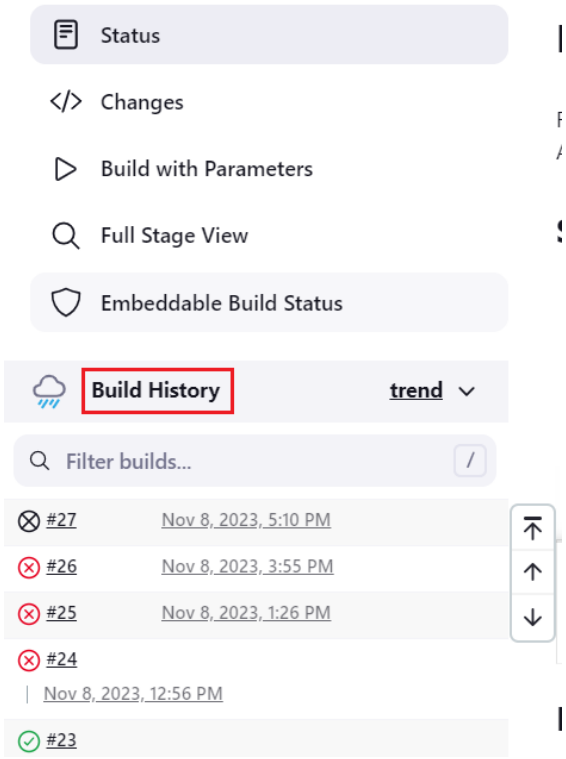


19. Dins de l' espai, podem comprovar els productes que s' han desplegat en el catàleg seleccionat.

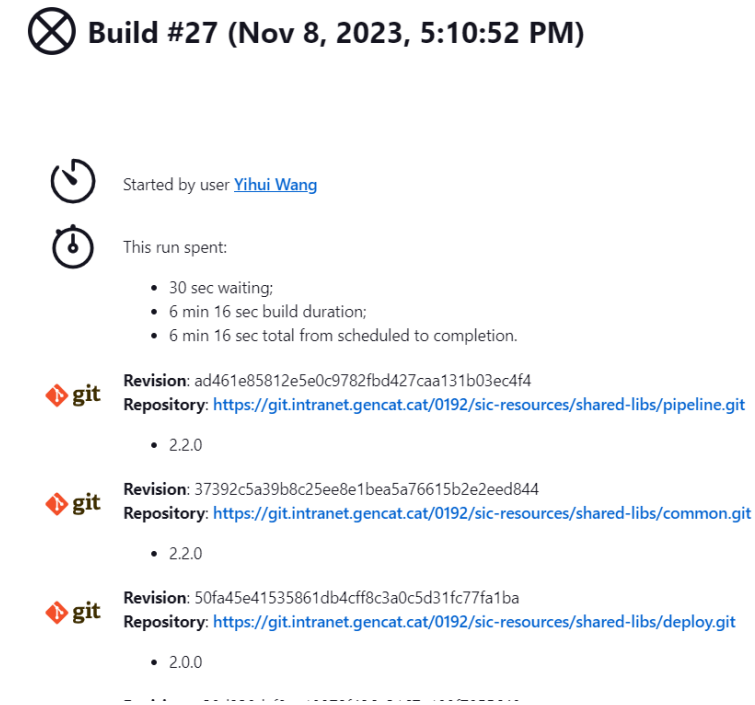


3.1.4.3 Consultar logs

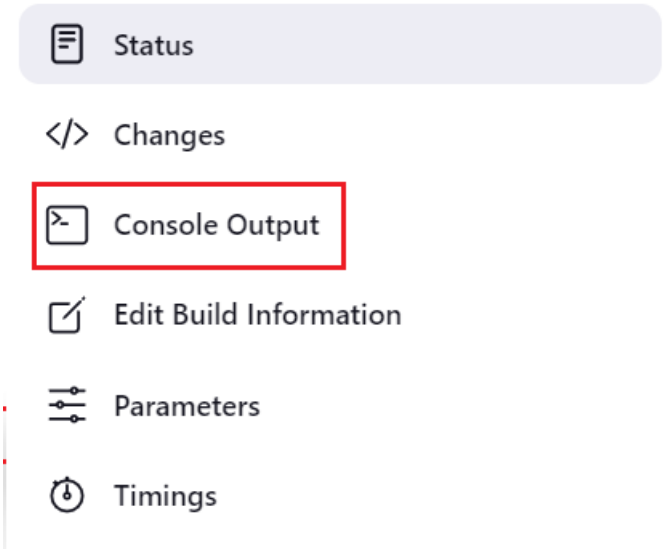
Per consultar els logs de pipeline, accedim al "Build History" al menú de l'esquerra.



Seleccionem ara el número d'execució que vulguem investigar per consultar els detalls de l'execució.



En aquesta secció es mostren els informes com el temps d' execució, l' iniciador de l' execució, i la imatge del pipeline que s' està usant. Després per consultar el log, accedim a la secció "Console Output".



En aquesta secció es veurà amb detall tots els logs que s' han generat al llarg de l' execució, juntament amb la informació dels errors, si escau.

3.1.4.4 Comportament de les operatives REPLACE, RETIRE i DELETE respecte als drafts

Amb un nou evolutiu corporatiu del pipeline, la responsabilitat de neteja del draft del Producte s'ha desplaçat des de l'operativa DELETE cap a les operatives REPLACE i RETIRE, on és conceptualment més coherent amb el cicle d'alta/baixa de versions del Producte.

- REPLACE: un cop l'operativa finalitza amb èxit, el pipeline esborra el draft del Producte sortint. Si existeix un fitxer map.yaml al repositori, s'aplica com a mapping file de plans (vegeu apartat MAP de 3.1.2).
- RETIRE: un cop l'operativa finalitza amb èxit, el pipeline esborra el draft del Producte retirat.
- DELETE: ja no executa neteja addicional de drafts; aquesta responsabilitat queda a càrrec dels REPLACE/RETIRE previs.

Cap de les tres operatives toca els drafts de les APIs referenciades pel Producte, perquè poden estar associades a tercers Productes del catàleg. La neteja eventual de drafts d'APIs orfes queda a càrrec de l'OFT amb eines de suport.

El job de neteja és idempotent: si el draft del Producte ja no existeix al subsistema de gestió, el pipeline continua sense fallar. Això permet reexecucions segures.

3.2 SIC+

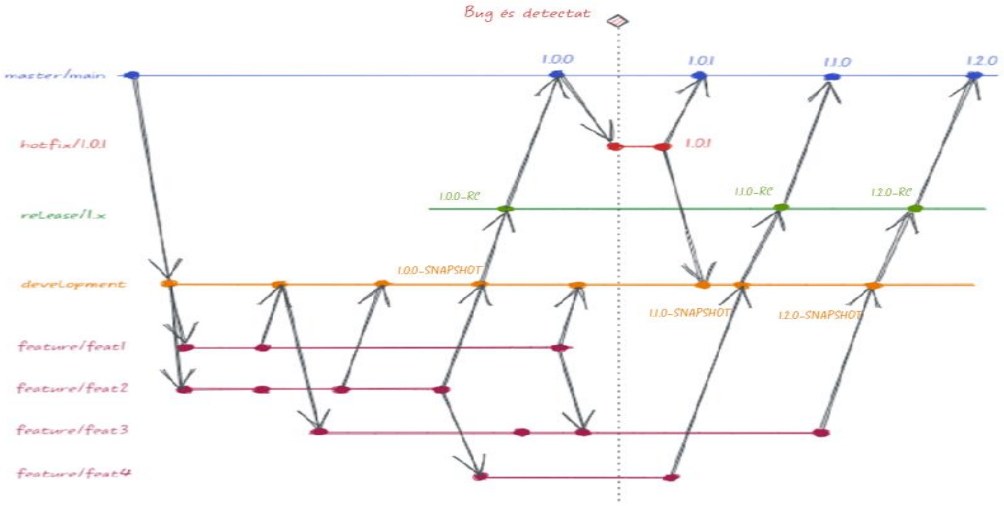
3.2.1 Consideracions prèvies dins del CTTI

L' organització de catàlegs, espais i productes és la següent:

- Actualment CTTI compta amb quatre catàlegs segons el tipus d' entorn i el tipus de visibilitat:
    - ***privat-pre*** i ***privat*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a la **Intranet**.
    - ***public-pre*** i ***public*** serien els entorns de preproducció i producció per desplegar en els catàlegs corresponents a l' **Internet**.
- Cada codi de diàleg disposarà d'un espai propi amb la nomenclatura "CD" + <codi_diàleg>. Per exemple: "CD0192".
- Un producte és una agrupació d'APIs i plans que les acompanyen (unitat mínima a versionar, desplegar i subscriure).

Un cop fet el desenvolupament amb API Designer, caldrà exportar els YAMLs de definició del producte i els seus APIs per dipositar-los en el repositori adequat de GitHub:

- Cada producte es correspondrà amb un repositori de GitHub, la nomenclatura de la qual és **<CODI_DIÀLEG>.<COMP>-<ACRÒNIM_APP>-<COMPONENT_TÈCNIC>-<TIPUS_COMP>**. Més informació a https://canigo.ctti.gencat.cat/plataformes/ghec/gh-model-govern/
- Es farà ús de la metodologia Gitflow per a la gestió de branques, havent de pujar-se el codi desenvolupat primer a una branca **feature**. Més informació aquí: https://canigo.ctti.gencat.cat/plataformes/ghec/modelTreball/model-gitflow-gitops/



En cas que el projecte no pugui adaptar-se a aquest model, caldrà realitzar una

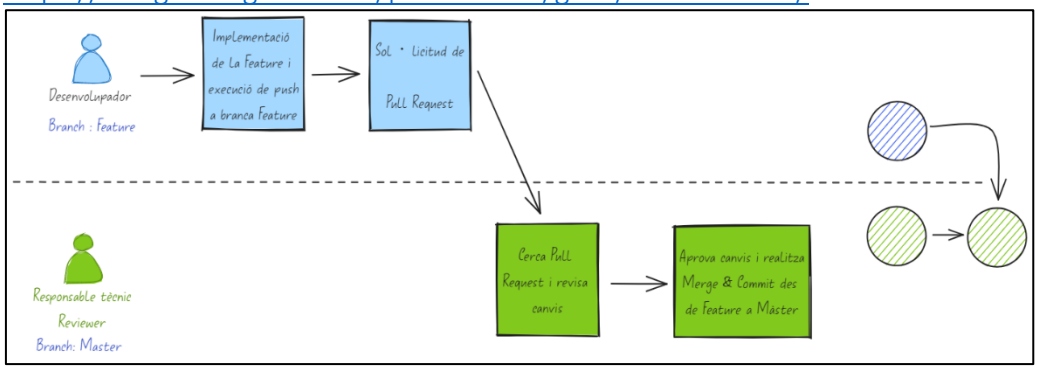
petició a perquè sigui estudiada la nova variant i veure'n la viabilitat. Addicionalment, i encara que s'han desvinculat les branques proposades amb els entorns disponibles per al desplegament, els workflows de CD disposen d'un pas anomenat "Env Matrix" que realitza validacions per discernir si un artefacte pot ser desplegat en un entorn en base al seu etiquetat (generat en el workflow de CI).

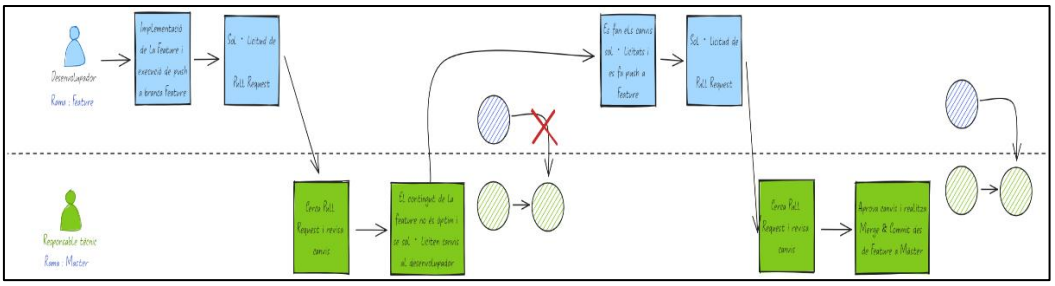
La política actual es mostra en el diagrama següent:

Entorn / Branch	Desenvolupament	Preproducció	Producció
Snapshot	✓	✗	✗
Release Candidate	✓	✓	✗
Final (sense prefix)	✓	✓	✓

Aquesta matriu pot ser modificada en base a les decisions que es prenguin segons les experiències o necessitats que vagin sorgint.

- El model de treball serà amb **Pull Request a GitHub**, metodologia en la integració de branques en la qual intervenen dos actors: el **desenvolupador** i el **responsable tècnic/Release Manager**. Aquest procés permet la revisió del contingut d'una branca **Feature** a integrar en una principal abans de realitzar el **Merge**. Si el contingut d'aquesta branca no és apte pels criteris de la persona que tingui el rol de revisor, aquesta branca no s'integrarà i se li demanarà al desenvolupador que solucioni els problemes trobats. Aquest model aposta per la qualitat i la detecció d'errors quan abans millor (shift-left). Tota la informació sobre el model de treball es pot trobar en el següent enllaç: https://canigo.ctti.gencat.cat/plataformes/ghec/modelTreball/





Aquest model de treball és de caràcter obligatori, i prova d'això és que les branques principals **Develop**, **Release** i **Master** estaran bloquejades per realitzar integracions directes sense l'ús de Pull Request.

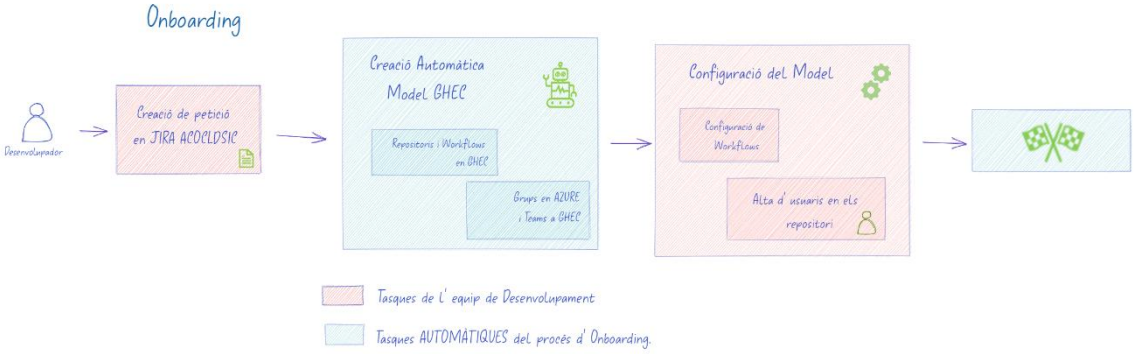
- Dins d'aquest model de desenvolupament, el versionat i tags d'artefactes estarà bloquejat al desenvolupador, de tal manera que seran els workflows automàtics de CI els que s'encarreguin d'aquesta tasca, permetent al desenvolupador només el versionat del seu codi font.
  S'ha instaurat el model **Semantic Version 2.0**. Tota la documentació relacionada es pot consultar en el següent link Model de Tag i versionat.

  Es podrà redesplegar en entorns preproductius, sobreescrivint el tag d'aquest entorn, però no així en **Producció**, on només es podrà desplegar una vegada correctament.

3.2.2 Configuració

Abans de començar a pujar el codi al repositori i executar els workflows que realitzaran els desplegaments, s' ha d' haver configurat tot el necessari a GitHub en base al nou model **GHEC**. Tota la informació es pot trobar al següent enllaç, https://canigo.ctti.gencat.cat/plataformes/ghec/gh-adopcio-model-ghec/, però aquí es deixa un resum:

El procés d'integració actualment és el següent:



3.2.2.1 Onboarding del projecte

Per tal d'integrar una nova aplicació al nou model de CI/CD a cloud públic, s' haurà d' especificar la següent informació:

Informació General

Camp	Exemples
Codi de Diàleg	“3292”
Codi de Component	“00”
Código de Departamento	“PRE”
Codi d’ Entitat	“CTT”
Lot de Manteniment	“AM22_23”
Acrònim aplicació	“apimanager”
Entorns a crear	“pre,pro”
Proveïdor Cloud	“azure”

Llistat de repositoris

Nom tècnic component	Tipus de Repositori	Categoria	Engine	Tecnologia	Versió
NomTecComponent6	apim	apim	N/A	apim	N/A

Exemple: amb taules anteriors, generaran el repositori anomenat 3292.00-apimanager-NomTecComponent6-apim.

L’alta d’una nova aplicació seguirà el flux normal d’Integració de Solucions (ISOL), a partir de la qual arribarà el tiquet pertinent a l’equip de Suport Cloud/SIC. Més detall a https://canigo.ctti.gencat.cat/plataformes/public_cloud/suport/comunicacio-suport-cloud/. És important que la taula de components vingui especificada en el Document d’Arquitectura (DA) de la solució a integrar.

Addicionalment als repositoris pels components que es demanen, se’n crearà un automàticament per a propòsits de Testing.

Un cop rebuda la petició, es processarà per part dels equips pertinents i, mitjançant un procés automàtic, es crearan els components i/o recursos necessaris a GitHub Enterprise Cloud i a Azure.

Addicionalment als repositoris pels components que es demanen, se'n crearà un automàticament per a propòsits de Testing.

3.2.2.2 Configuració del model GHEC

Un cop ha finalitzat el procés automàtic que crea tots els recursos necessaris per donar suport al model, s'han de realitzar les següents configuracions bàsiques:

1. Alta d'usuaris en grups depenent del Rol
   El procés automàtic de creació del model crea automàticament un conjunt de grups d'Azure Entra ID que s'utilitzaran per assignar rols a usuaris.

   Els grups que es creen depenen del departament, entitat i lot de manteniment de l'aplicació (exceptuant els transversals). Si el team ja existeix prèviament, no es tornarà a crear:

   - sec-read: Equip transversal de Seguretat (ACC).
   - qa-read: Equip transversal de Qualitat (CTTI).
   - srv-read: Equip transversal de Servei (CTTI).
   - arq-read: Equip transversal d'Arquitectura (CTTI).
   - </departament/>-</entitat/>-</lot_manteniment/>-maintain: Leaders tècnics del lot de manteniment.
   - </departament/>-</entitat/>-</lot_manteniment/>-write: Desenvolupadors del lot de manteniment.
   - </departament/>-</entitat/>-read: Per a Gestors de Solucions / Entrega de l'àmbit.

   Una vegada creats els grups, cal que l'owner o owners identificats donin d'alta els diferents usuaris en els grups pertinents depenent del rol que han de realitzar (Maintain o Write).

   Al següent enllaç es detalla com és la gestió d'usuaris, a més de com es gestionen les llicències de GitHub Enterprise Cloud –> Gestió d'usuaris i llicències

2. Configuració inicial workflows
   El workflow cridant necessita configurar una sèrie de parametres per al workflow anomenat.

   Aquests paràmetres estan explicats a la següent documentació Configuració workflows.

3. Configuració inicial per a invocacions ITSM

   Dins dels diferents workflows de CD, existiran steps encarregats de realitzar invocacions a ITSM (Remedy) en les quals es crearan WorkOrders on s'indicarà que el sistema està realitzant un desplegament d'una aplicació i l'estat final d'aquest desplegament.

   Per comunicar-nos amb la plataforma ITSM serà necessari configurar el fitxer **itsm-inputs.json**, disponible al repositori, on s'indicaran les característiques del tiquet que es crearà en base a la naturalesa del desplegament.

- **“ITSM_SERVICE”:** Servei Remedy que realitza el desplegament (l'associat al vostre codi de diàleg, no el API MANAGER). És important afegir exactament el valor donat d’alta en Remedy.
- **“ITSM_CLASS”:** Podrà tenir els següents valors, depenent del tipus de desplegament:
    - Normal
    - Emergency
- **“TYPE_AFFECTATION”:** Podrà tenir els següents valors:
    - DEGRADACIO
    - SENSE TALL DE SERVEI
    - TALL DE SERVEI
- **“ITSM_AFFECTATION”:** Text lliure per indicar el motiu del desplegament
- **“ITSM_IMPACT”:** Per indicar l’impacte que pot tenir el desplegament, amb els possibles valors:
    - 4-Minor/Localized
    - 3-Moderate/Limited
    - 2-Significant/Large
    - 1-Extensive/Widespread
- **“ITSM_TIER3”:** Nivell 3 de categorització Remedy d’un tiquet que podrà tenir els següents valors:
    - ADAPTATIU
    - CORRECTIU
    - EVOLUTIU
    - PREVENTIU
    - VERSIO
    - VULNERABILITAT
- **“ITSM_PRIORITY”:** Prioritat del desplegament amb els següents valors:
    - 4-Low
    - 3-Medium
    - 2-High
    - 1-Critical

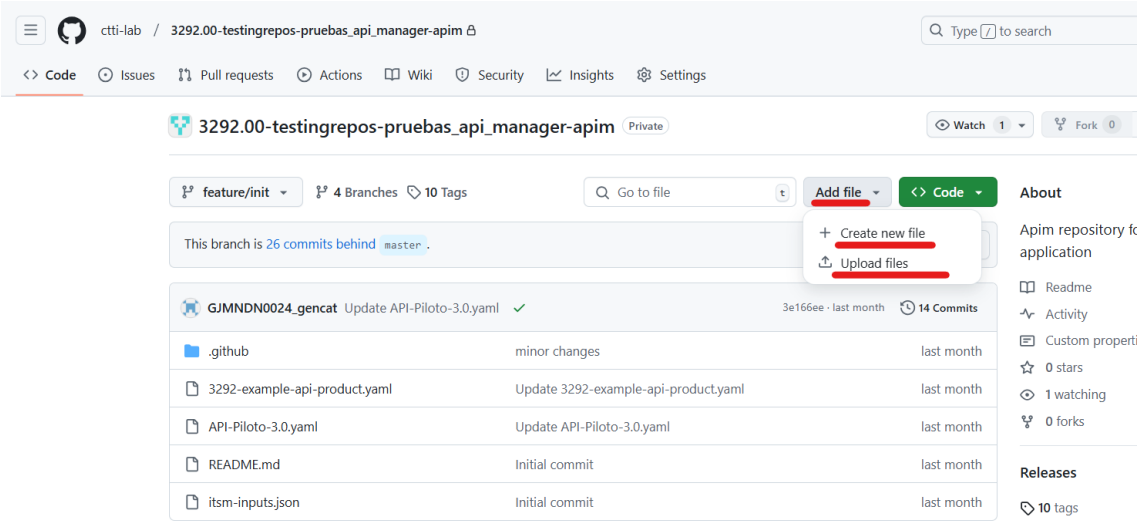
3.2.3 Pujada de carpetes i fitxers al repositori

Una vegada finalitzades les fases de disseny tant de les APIs com dels productes, on s' han configurat els fitxers YAML corresponents a cadascun d' ells, s' hauran de pujar aquests fitxers al repositori assignat al projecte, per posteriorment procedir amb el

desplegament entre entorns. Els fitxers del producte i els seus corresponents APIs han de trobar-se en la mateixa ruta, la ruta principal del repositori.

Tal com s' ha esmentat abans, es fa ús del model Gitflow de branques. En el següent link Model GitFlow, es mostra l'article amb els models a seguir, juntament amb les branques que entren en joc.

Es podran pujar aquests fitxers tant des de la consola web de GitHub com des de qualsevol client Git que hàgim configurat correctament.



3.2.3.1 map.yaml (arxiu opcional per a l'operativa REPLACE)

Es tracta d'un fitxer opcional que s'allotja al repositori del projecte, a la mateixa altura que els fitxers de productes i APIs (no en cap subcarpeta). El seu nom ha de ser exactament map.yaml. Si no existeix, s'aplica el comportament per defecte.
Aquest fitxer defineix el mapeig de plans entre el Producte sortint (l'antic, que se substitueix) i el Producte entrant (el nou) en l'operativa REPLACE. Segueix el format estàndard documentat per IBM per al mapping file del comandament apic products:replace:

```
plans:
  - source: <plan_sortint_1>
    target: <plan_entrant_1>
  - source: <plan_sortint_2>
    target: <plan_entrant_2>
```

Quan cal aportar-lo:

- Si els plans del Producte entrant tenen el mateix nom que els del sortint i el mapeig per defecte (source == target) cobreix tots els plans, el fitxer no és necessari.
- Si hi ha renombrats de plans, fusions (N source → 1 target), afegit de plans nous o qualsevol divergència de plans entre Producte sortint i entrant, el fitxer és obligatori; en cas contrari, el REPLACE fallarà controladament.

Comportament del pipeline:

- Sense map.yaml i plans coincidents → REPLACE OK; les subscripcions migren del plan source al plan target corresponent (source == target).
- Sense map.yaml i plans no coincidents → el pipeline falla controladament amb un missatge clar.
- Amb map.yaml correcte → REPLACE OK; les subscripcions actives migren del plan source al plan target indicat.
- Amb map.yaml incomplet o referenciant un target inexistent al Producte entrant → el pipeline falla controladament amb un missatge clar.

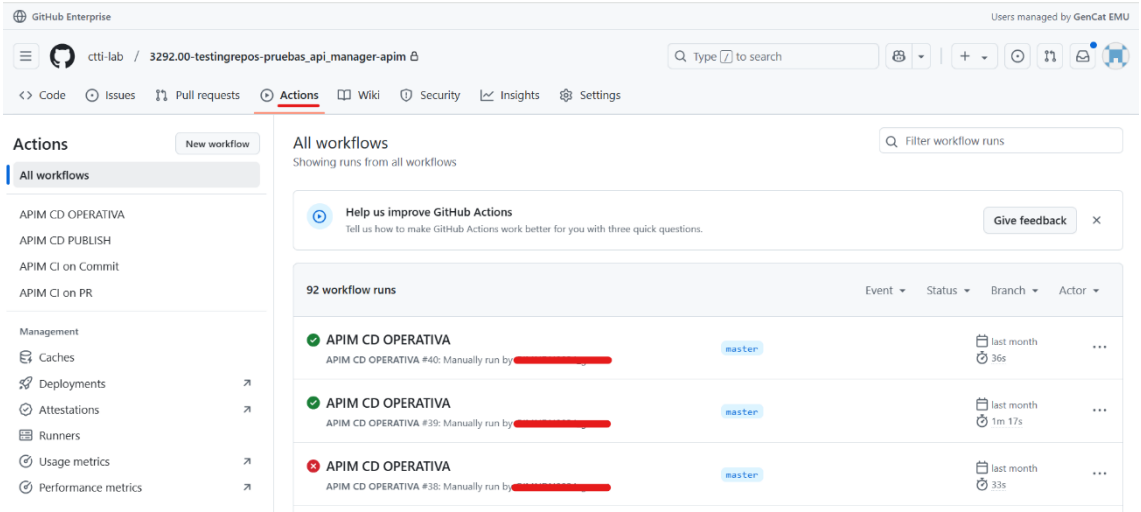
Exemple:

```
plans:
  - source: default-plan
    target: bronze
  - source: silver
    target: silver
  - source: gold
    target: new-plan
```

3.2.4 Desplegament utilitzant workflows

Per a cadascun dels repositoris creats, es creen un conjunt de workflows que seran els que executin les tasques de CI/CD (són els anàlegs a les pipelins de Jenkins de SIC 3.0).

L'accés a aquests workflows es realitzarà a través de l'opció "Actions" de cada repositori a GHEC.



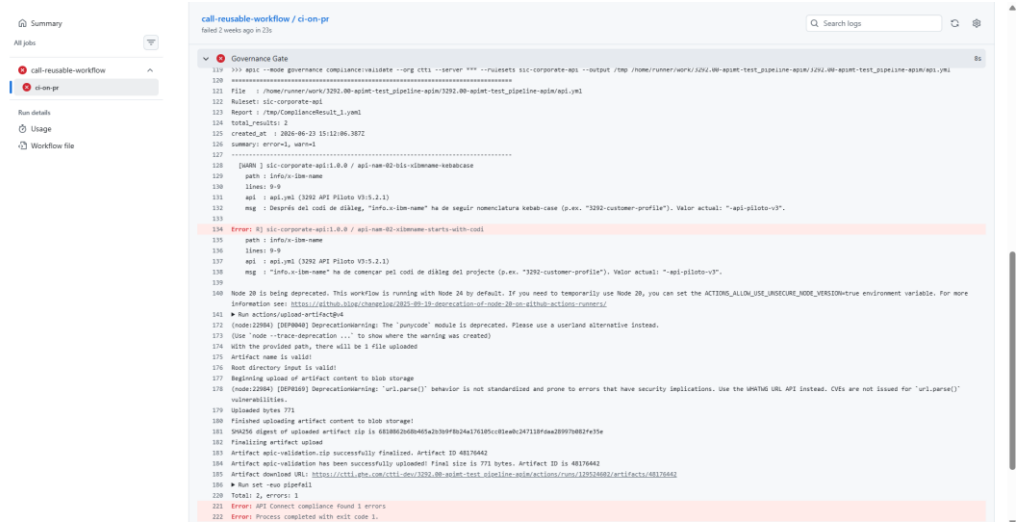
L'execució dels workflows dependran de la seva tipologia i del model definit, essent:

- ***Workflows de CI:*** Executats automàticament en la sol·licitud d'un Pull Request o en l'execució d'un Merge de dita Pull Request. S'ha diferenciat en el workflow entre ***canvis en temps de Pull Request (PR)***, que equivaldria al procés pel qual un usuari crea la PR, i encara no és validada per un moderador o usuari del repositori i ***canvis en temps de Commit***, que equivaldria al procés després d'haver-se acceptat la PR, i integrar ambdues branques involucrades.
    - ***APIM CI on PR.*** Aquest workflow, depenent d'aquestes branques que es vulguin "mergear", executarà diferents steps amb diferents jobs. Si es crea una PR d'una branca ***feature*** a la branca ***develop***, en temps d'execució es

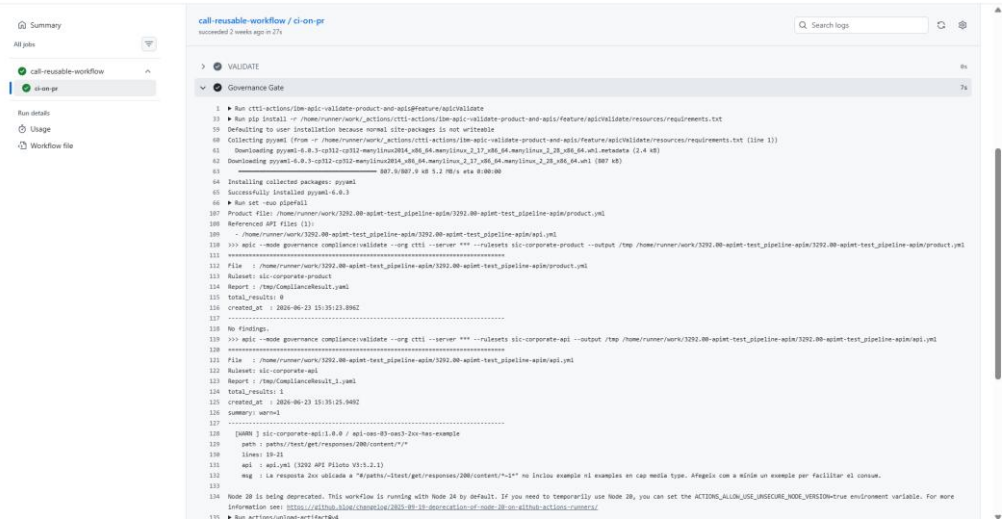
llançarà el workflow de CI que executarà els steps de validació de codi apic i gobernança i aquest resultat es comenta en la PR.

El workflow APIM CI on PR executa, com a part de la fase de validació de codi, una validació de governança contra els rulesets corporatius sic-corporate-api i sic-corporate-product. El resultat queda registrat com a comentari a la Pull Request:

- Si la validació detecta findings de severitat error, la PR no es pot mergear: la branca queda bloquejada fins que es corregeixin els findings.

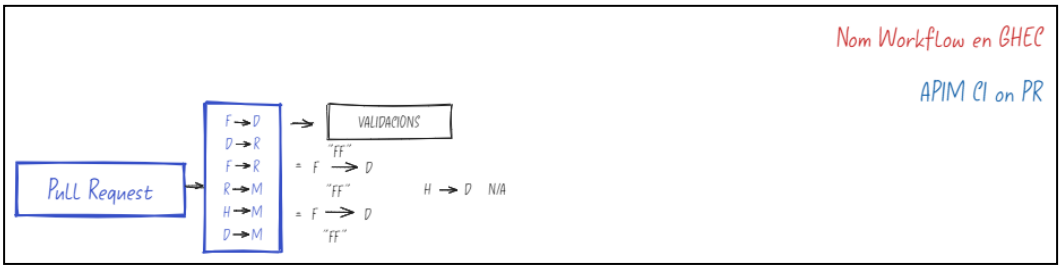


- Si només hi ha findings de severitat warn, la PR es pot mergear, però els avisos queden registrats com a recomanació al comentari del workflow.

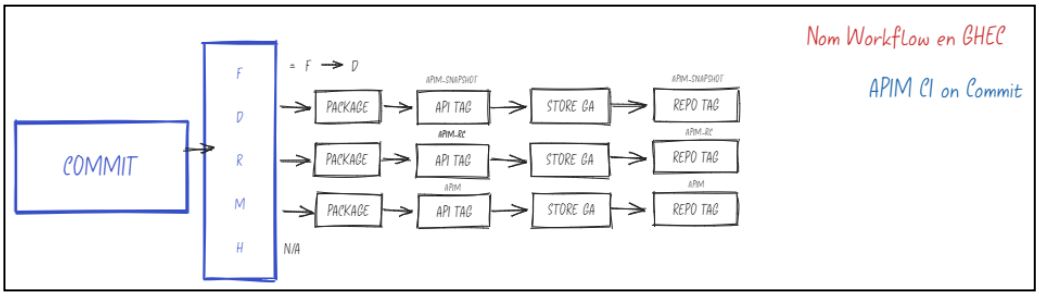


La descripció funcional completa de les regles és al Disseny tècnic i a l'ERQ corporatius dels rulesets sic-corporate-api / sic-corporate-product.

En canvi, si la PR es fes entre les branques ***develop-release***, ***release-master***, ***hotfix-master***, s'ometrien aquests steps i es realitzaria un fast-forward, ja que tots ells haurien estat executats i validats prèviament, donat que teòricament el codi no rep més canvis des que entra en la branca develop en endavant.

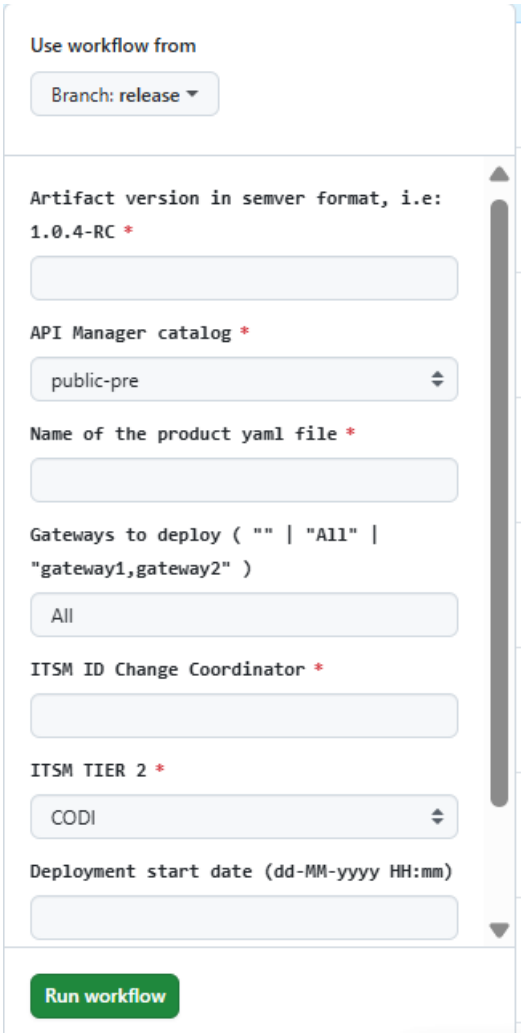


- ***APIM CI on Commit.*** Si estem en temps de commit, i partint de la base que el paquet no ha de ser mutable entre els diferents entorns, es comprova el tag generat en temps de PR, es publica l'artefacte a ***Github Artifacts*** i es torna a fer el tag del repositori.



- ***Workflows de CD:*** Executats sota demanda a través de la interfície web de GHEC.

- *APIM CD PUBLISH.* El workflow farà la publicació d'una nova versió d'un producte i APIs associades. El sistema permet redesplegar versions als catàlegs preproductius sempre que no hagin arribat a producció.



  - **Input opcional gateways.** El workflow accepta un input opcional gateways que permet restringir el desplegament a un subconjunt dels gateways associats a l'espai destí. La semàntica és la mateixa que la de la variable APIC_GATEWAY_SERVICES de l'aca.yml de SIC 3.0:
    - Si l'input s'omet, val "All" o és una cadena buida → desplegament a tots els gateways associats a l'espai (comportament per defecte; retrocompatible al 100%).
    - Si l'input declara una llista no buida → desplegament restringit als gateways indicats.

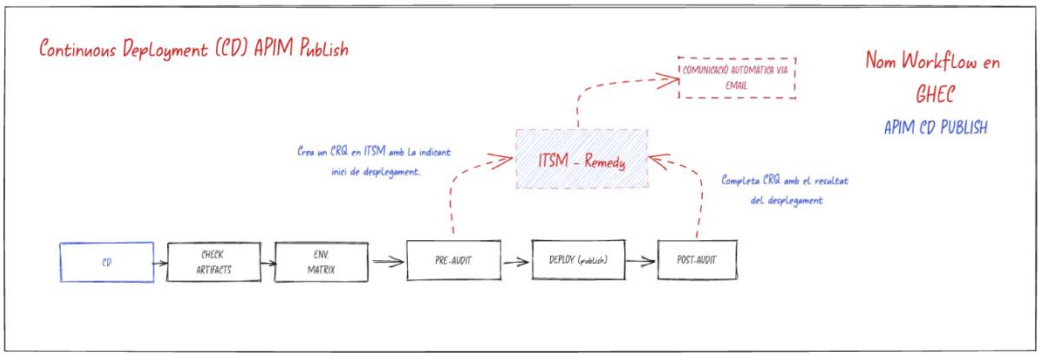
    Si algun dels gateways indicats no existeix a l'espai destí, el workflow falla controladament abans d'iniciar cap publish, sense desplegament parcial.
    Aquest paràmetre vindrà ja configurat en els nous repositoris creats, però si el teu repositori va ser creat abans del 15 de setembre 2026, hauràs de modificar els workflows existents de

PUBLISH per configurar-lo d'aquesta manera, afegint on correspon:

```
on:
  workflow_dispatch:
    inputs:
      gateways:
        description: 'Gateways destí ( "All" | "gateway1,gateway2" | "" )'
        required: false
        default: 'All'

# Crida al workflow reusable
jobs:
  call-reusable-workflow:
    uses: ctti-arq/reusable-workflows/.github/workflows/apim-cd-publish-
reusable.yaml@v5

    secrets: inherit
    with:
      gateways: ${{ inputs.gateways }}
```



on:

- ***CHECK ARTIFACTS:*** Comprova l'existència i validesa de l'artefacte en el repositori.
- ***ENVIROMENT MATRIX:*** Valida si l'artefacte pot ser desplegat en l'entorn especificat.
- ***ITSM PRE AUDIT:*** Realitza una auditoria prèvia en ITSM creant una CRQ per a la preparació del desplegament (només en entorns diferents de dev).
- ***PUBLISH PRODUCT TO IBM API MANAGER (Deploy):*** Publica el producte a IBM API Manager segons els paràmetres configurats.
- ***ITSM POST AUDIT:*** Completa l'auditoria en ITSM després del desplegament, registrant l'estat final i completant la CRQ.

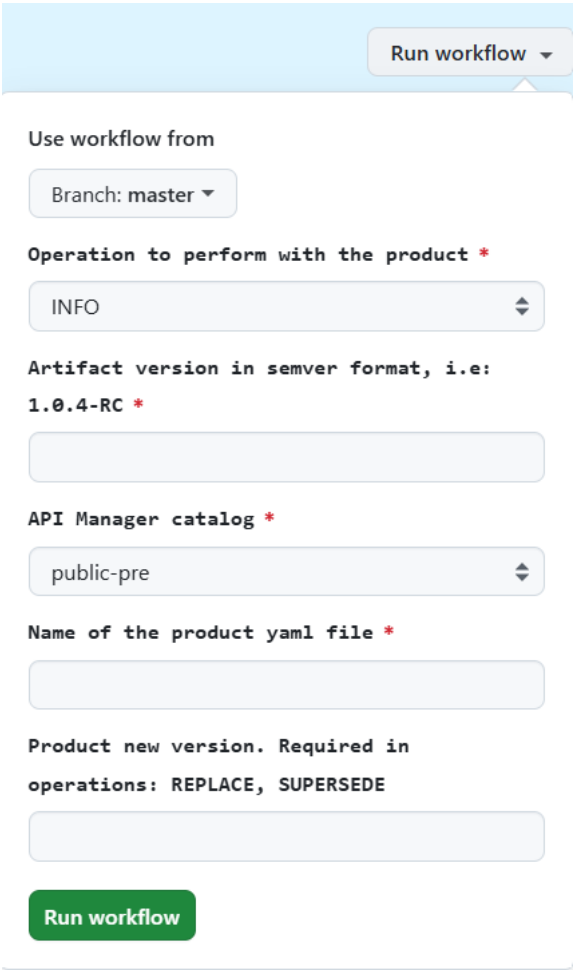
- ***APIM CD OPERATIVA.*** El workflow realitzarà una de les següents operatives: ***INFO***, ***DELETE***, ***DEPRECATE***, ***RETIRE***, ***REPLACE***, ***SUPERSEDE***.

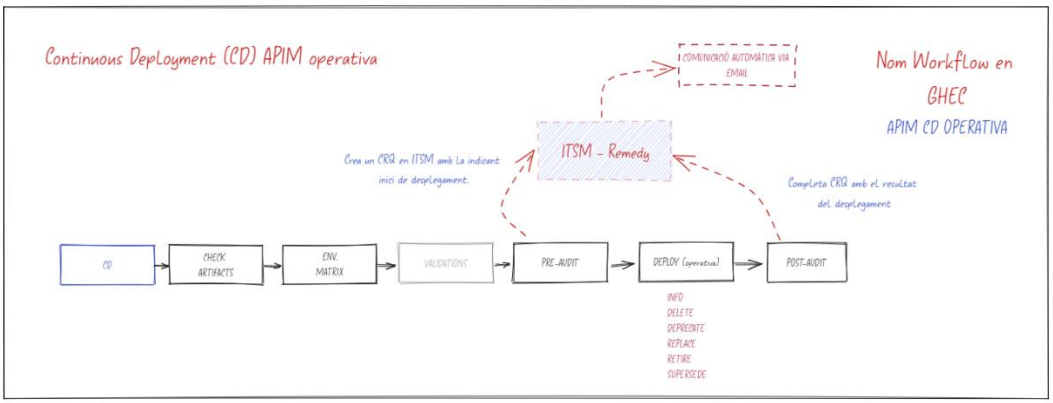
Comportament específic per operativa amb l'evolutiu del pipeline.

- REPLACE: el workflow detecta automàticament el fitxer map.yaml del repositori (a la mateixa altura que els fitxers de productes i APIs) i l'utilitza com a mapping file de plans. Vegeu apartat MAP de 3.2.3.1 per al format i el comportament del fitxer. Un cop REPLACE finalitza amb èxit, el workflow esborra el draft del Producte sortint.
- RETIRE: un cop l'operativa finalitza amb èxit, el workflow esborra el draft del Producte retirat.
- DELETE: ja no executa neteja addicional de drafts; aquesta responsabilitat queda a càrrec dels REPLACE/RETIRE previs.

Cap de les tres operatives toca els drafts de les APIs referenciades pel Producte. La neteja massiva de drafts d'APIs orfes queda a càrrec de l'OFT.

El job de neteja és idempotent: si el draft del Producte ja no existeix, el workflow continua sense fallar.





on:

- ***CHECKOUT:*** Realitza la verificació de codi en el repositori.
- ***CHECK ARTIFACTS:*** Verifica l'existència i validesa de l'artefacte al repositori.
- ***ENVIROMENT MATRIX:*** Valida si l'artefacte pot ser desplegat en l'entorn especificat.
- ***VALIDATIONS:*** Step preparat per a inserir les futures validacions a realitzar prèvies a l'execució de les operatives.
- ***ITSM PRE AUDIT:*** Realitza una auditoria prèvia a ITSM, creant una CRQ per al desplegament (només en entorns diferents de dev).
- ***PRODUCT OPERATION INTO IBM API MANAGER (Deploy):*** Executa operacions a IBM API Manager segons l'operació especificada.
- ***ITSM POST AUDIT:*** Completa l'auditoria a ITSM després del desplegament, registrant l'estat final i completant la CRQ.

Dins del repositori, el codi dels workflows estarà disponible en la següent ruta: <repositori>/.github/workflows

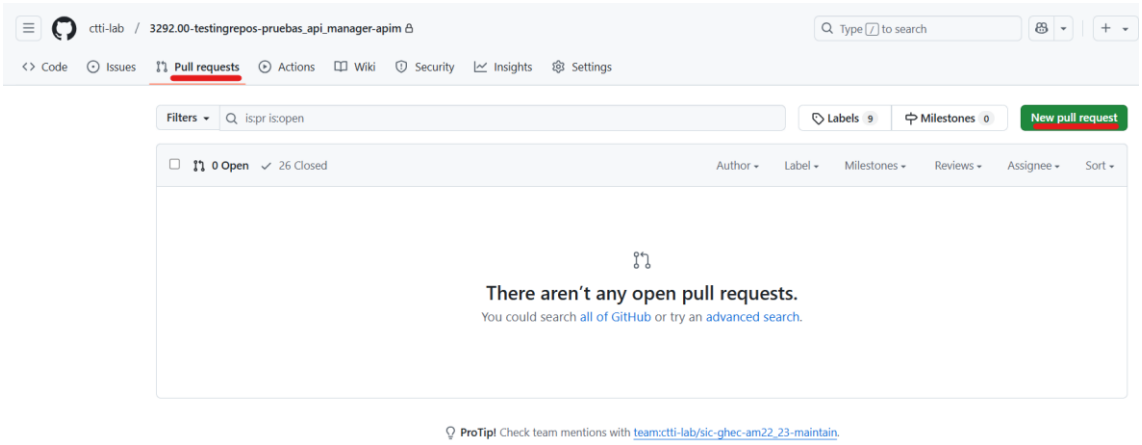
3.2.4.1 Passos per desplegar utilitzant workflows

Com ja s'ha comentat anteriorment, el nou model de treball es basarà en:

- Gestió de branques amb un model Gitflow.
- Integracions de branques basades en Pull Request.
- Tagging d'artefactes i repositoris amb el model Semantic Version 2.0.

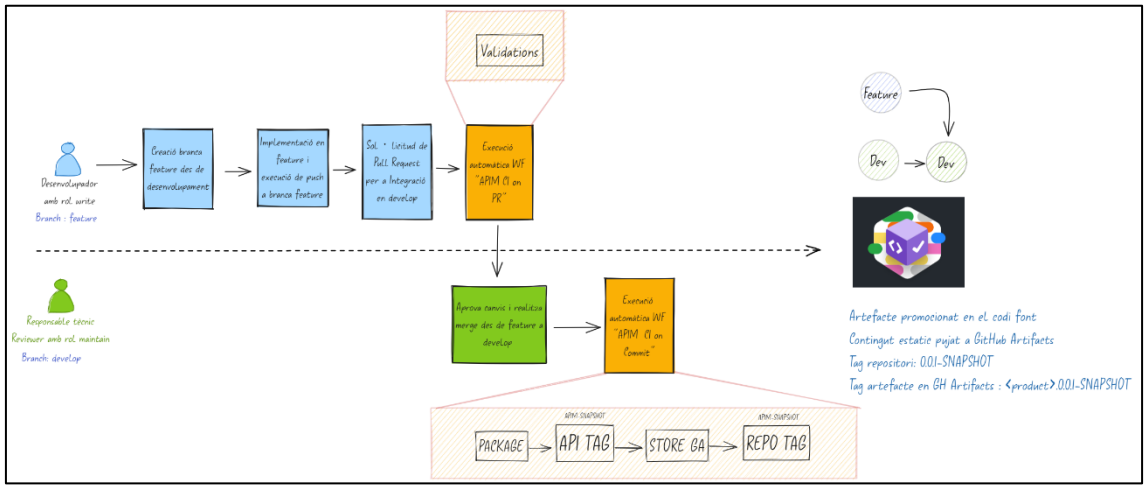
A continuació, es mostra l'execució e2e d'un flux de treball, des que el desenvolupador realitza la seva implementació de una API en una branca Feature, fins al desplegament en el catàleg Producció. En aquest cas, l'aplicació no disposa d'entorn de desenvolupament, **l' estratègia seria la integració de feature en develop i d' aquesta a rellegeixi sense desplegar en l' entorn de development**.

1. El desenvolupador ja ha realitzat el seu desenvolupament en la branca feature i es disposa a integrar els seus canvis a la branca develop. Per a això, realitza una Pull Request dels canvis a develop.

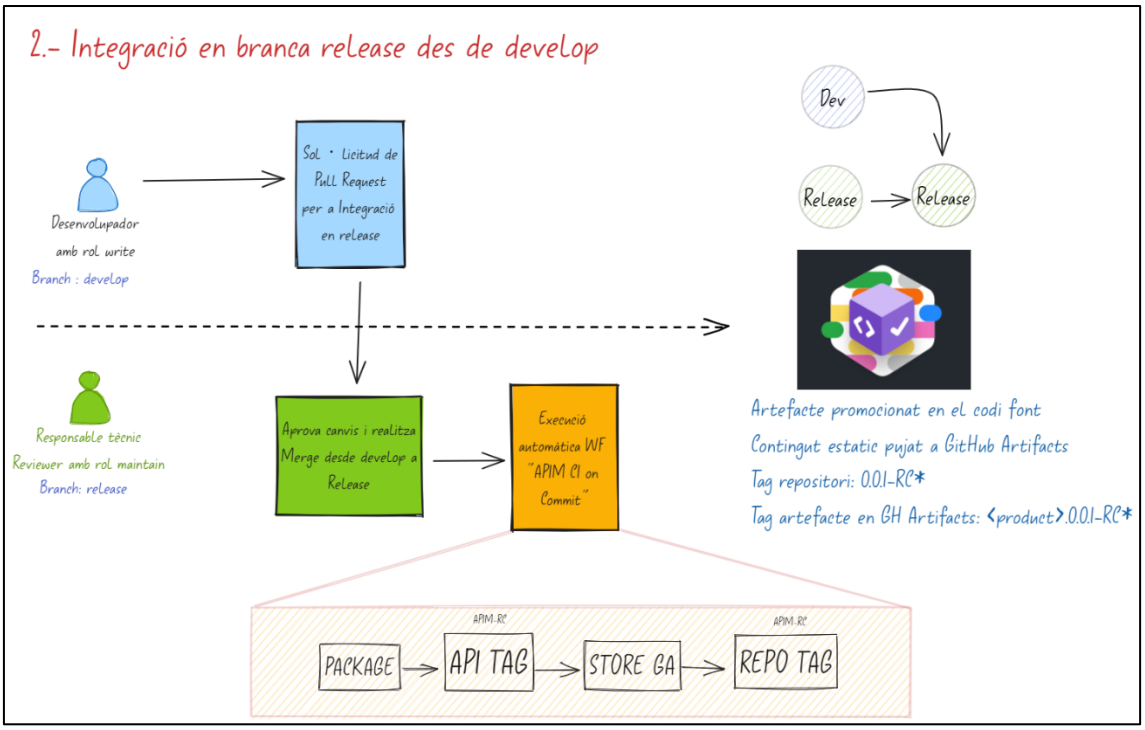


2. De manera automàtica, s'executa el workflow "APIM CI on PR", on es validarà el codi del YAML.

3. Un cop es validi, el responsable tècnic/Release Manager que tingui permisos de Maintainer, haurà de revisar la Pull Request i aprovar-la en cas que estigui tot correcte. Una vegada s'aprova, es llançarà de manera automàtica el workflow "APIM CI on Commit", el qual comprova el tag generat en temps de PR, publica l'artefacte a Github Artifacts i torna a fer el tag del repositori.

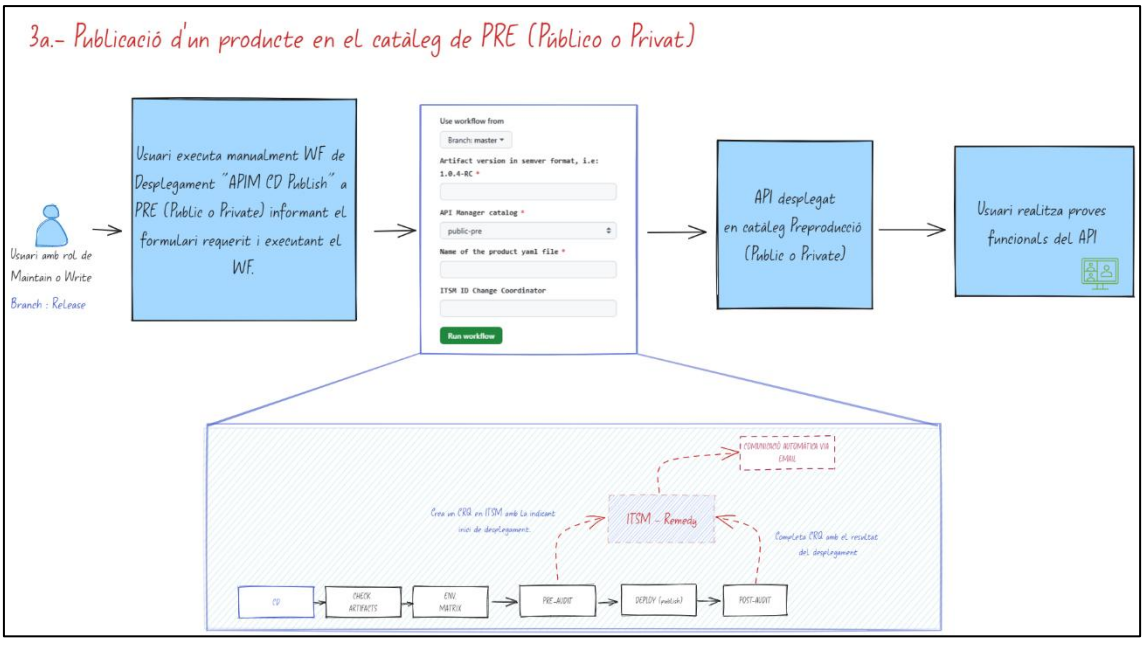


4. A continuació, el desenvolupador ha de realitzar una altra Pull Request per migrar el codi a la branca rellegint-se de cara a poder desplegar l' artefacte a l' entorn de PRE d' API Manager. En aquest cas, igual que en el pas anterior, el responsable tècnic/Release Manager ha d'aprovar la Pull Request de nou perquè torni a executar-se el workflow "APIM CI on Commit". Aquesta vegada no s'executarà el workflow "APIM CI on PR".

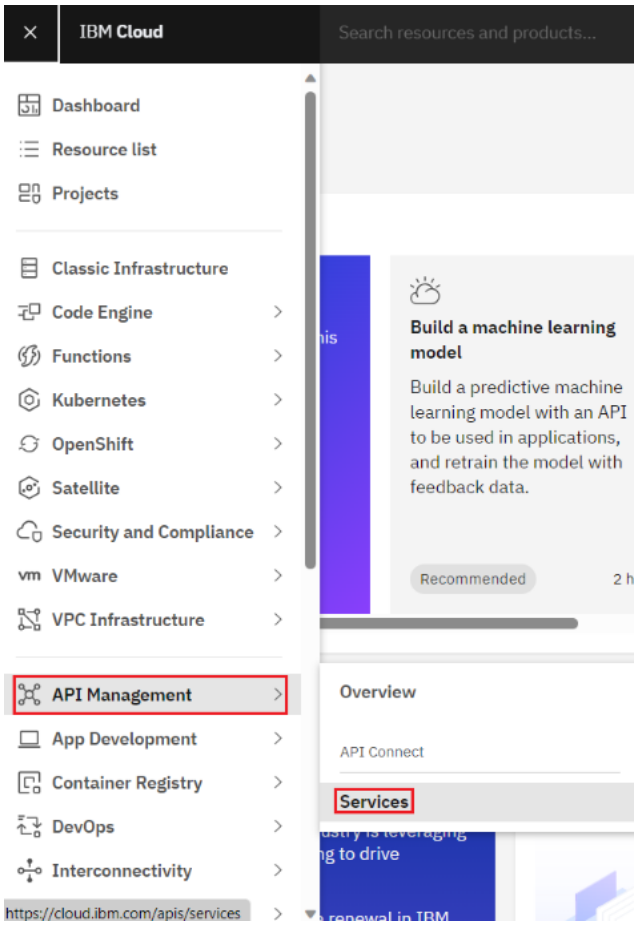


5. Després, el desenvolupador ja podrà executar de forma manual el workflow "APIM CD PUBLISH" per desplegar el producte guardat en l'artefacte de GitHub en l'entorn de preproducció de l'API Manager. Per a això, haurà d'accedir a aquest workflow a la secció d'"Actions", polsar a "Run workflow" i omplir el formulari.

   Els camps necessaris són la branca (el valor de la qual hauria de ser release), versió de l'artefacte (el valor del qual hauria de ser X.X.X-RC), el catàleg on es desplegarà (el valor del qual hauria de ser public-pre o privat-pre), el nom exacte del fitxer YAML del producte i l'ITSM ID Change Coordinator (el valor del qual hauria de ser ID de l'usuari per crear la CRQ en ITSM amb l'objectiu d'informar sobre el desplegament.).

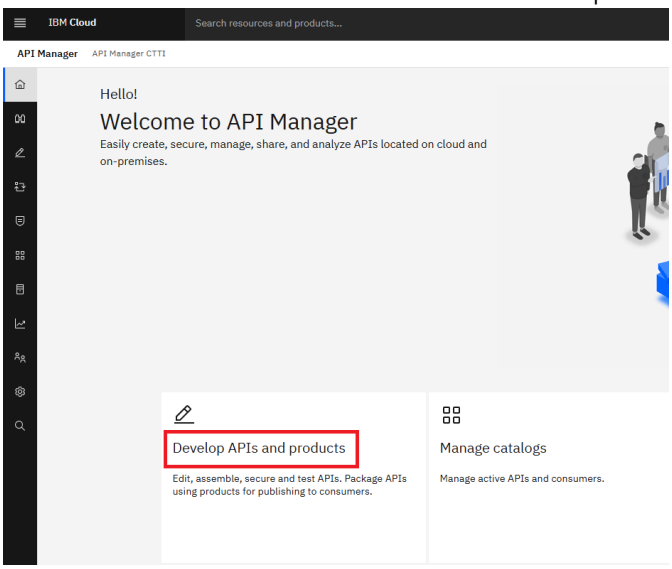


6. Després, per comprovar que el producte i l'api s'ha desplegat correctament, ens dirigim a https://cloud.ibm.com/authorize/gicar i ens identifiquem amb les credencials de GICAR.
7. Un cop dins, accedim al menú ubicat a l'esquerra i accedim a "API Management – Services".

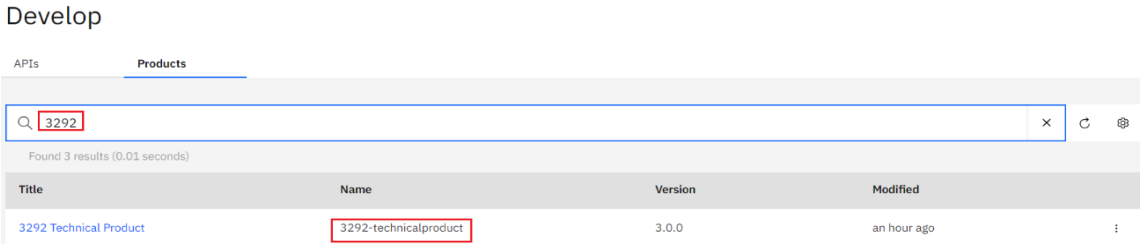


8. Ara accedeixen dins de CTTI.

9. Accedim a la secció "Develop APIs and products".

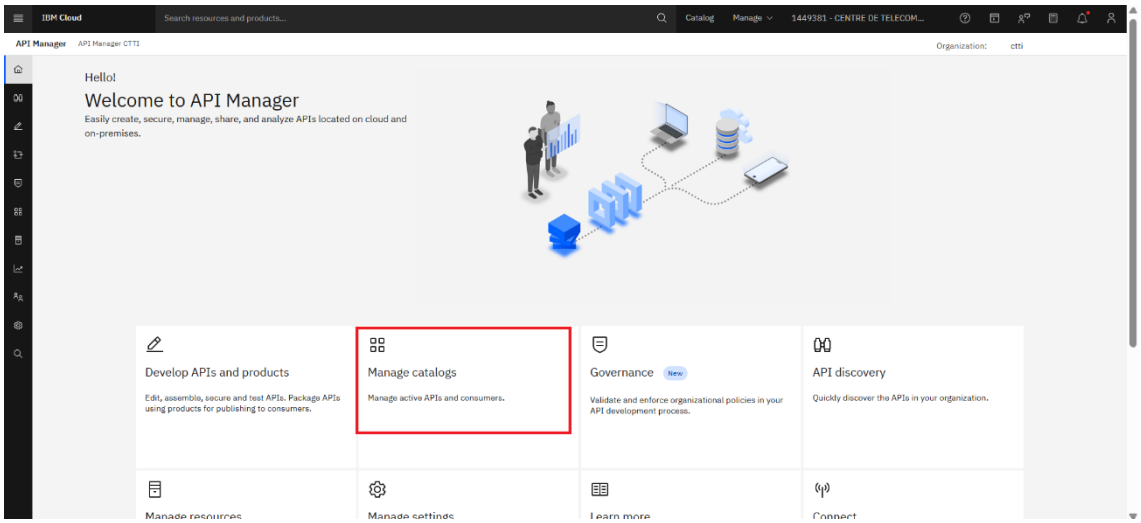


10. A la barra de recerca de productes, introduïm el nom del nostre producte desplegat i confirmem que s'ha desplegat correctament el draft del producte.

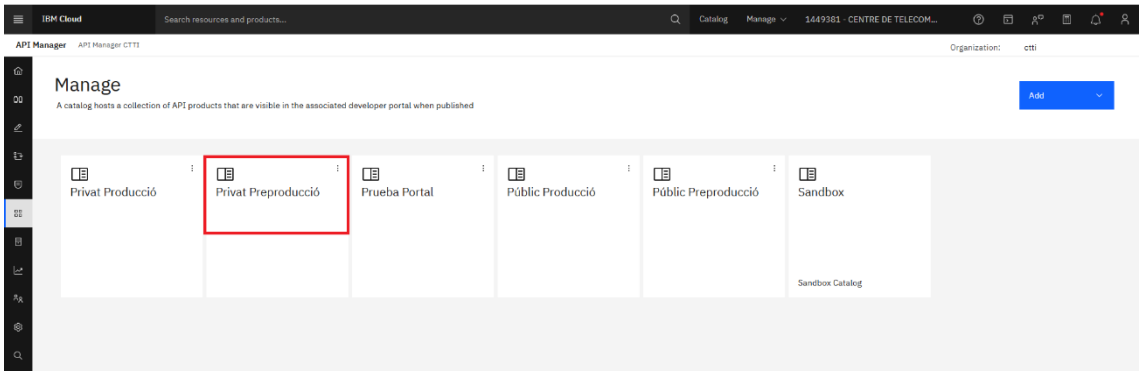


11. A la barra de recerca d'APIs, introduïm el nom de la nostra API desplegada i confirmem que s'ha desplegat correctament el draft de l'API.

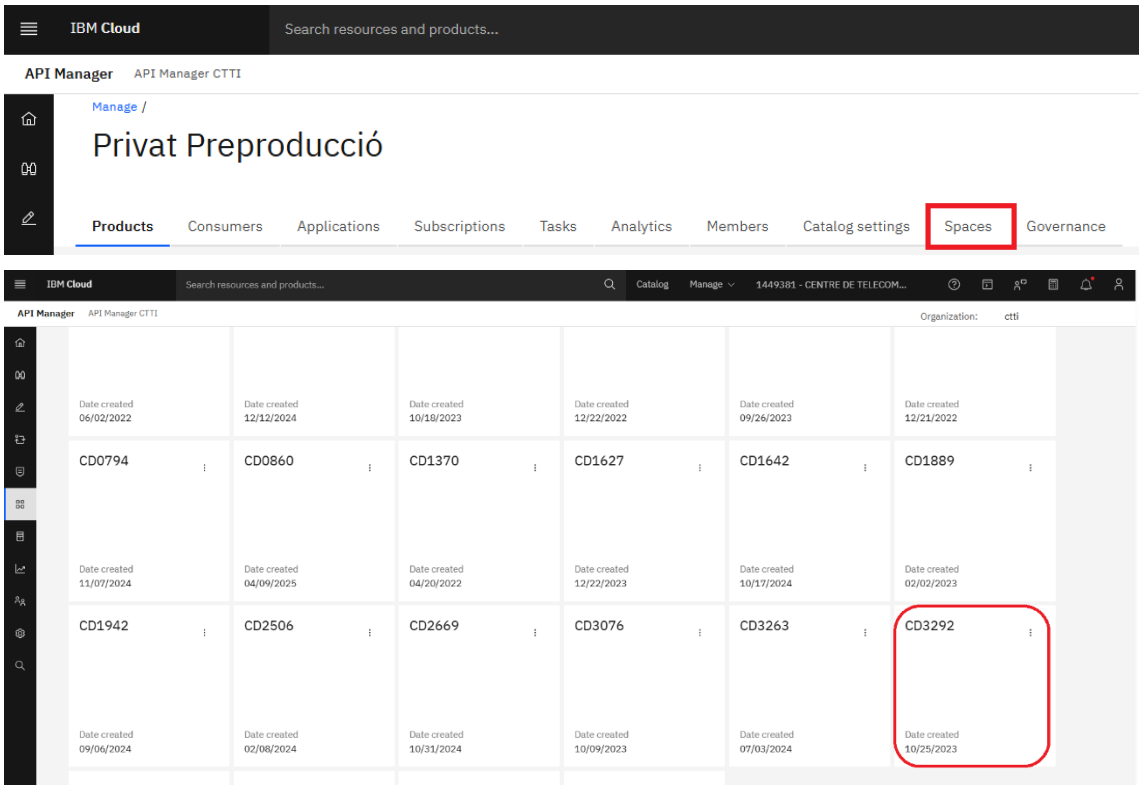
12. A continuació, comprovem que el producte s'ha desplegat al catàleg. Per a això, accedim a la secció "Manage Catalogs".



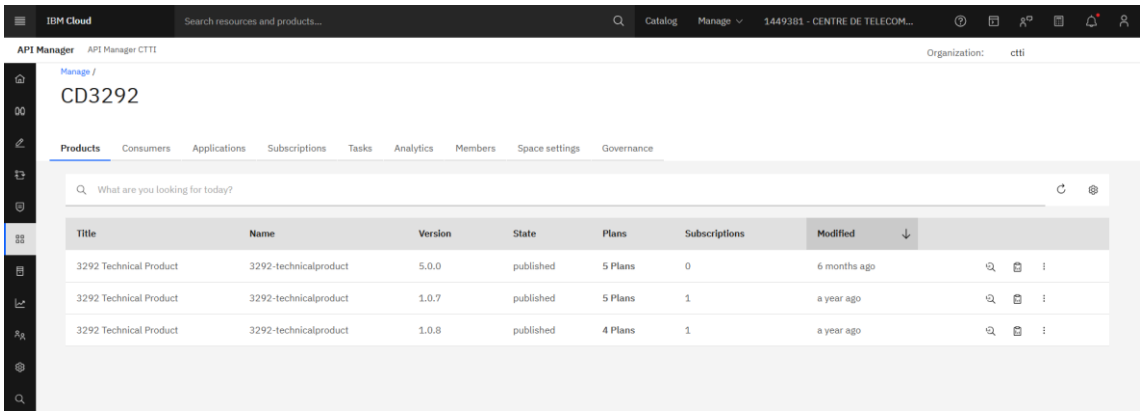
13. Accedim al catàleg per on s'ha desplegat el nostre producte, per exemple Privat Preproducció.



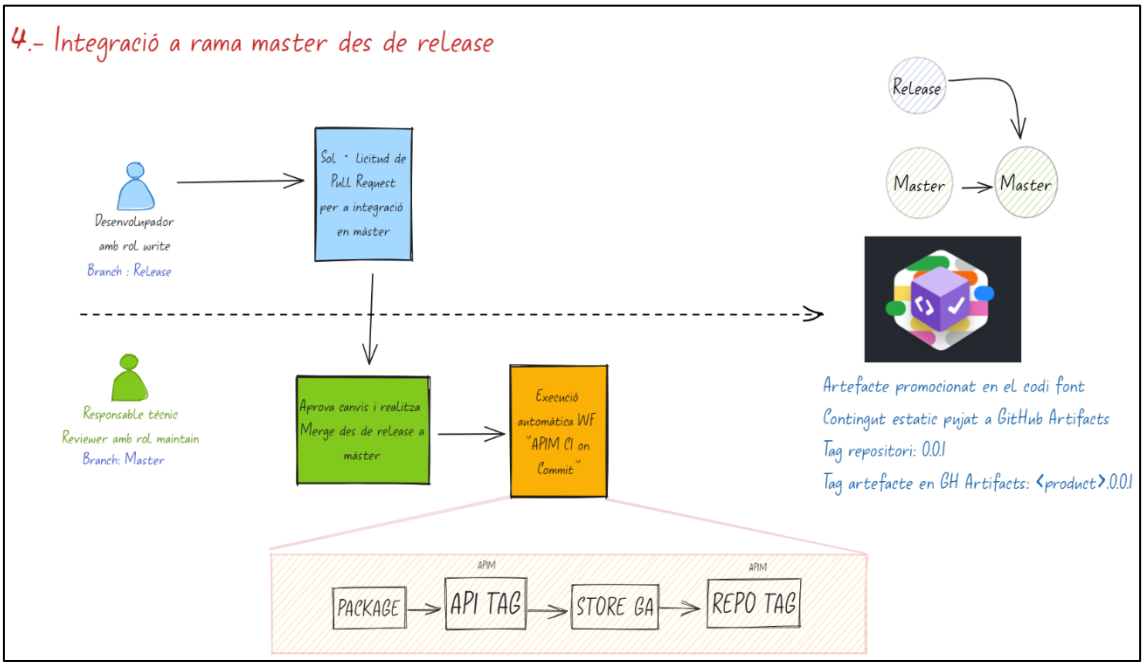
14. Ara entrem al nostre espai assignat dins del catàleg, per exemple CD3292.



15. Dins de l' espai, podem comprovar els productes que s' han desplegat en el catàleg seleccionat.



16. Un cop s' hagin fet proves correctament a l' entorn de Preproducció, es pot triar fer la pujada a Producció. Per a això, primer el desenvolupador tornarà a haver de realitzar una Pull Request de la branca rellegint-se a màster. Aquesta Pull Request haurà de ser aprovada de nou pel responsable tècnic/Release Manager. Tornarà a executar-se només el workflow “APIM CI on Commit”.

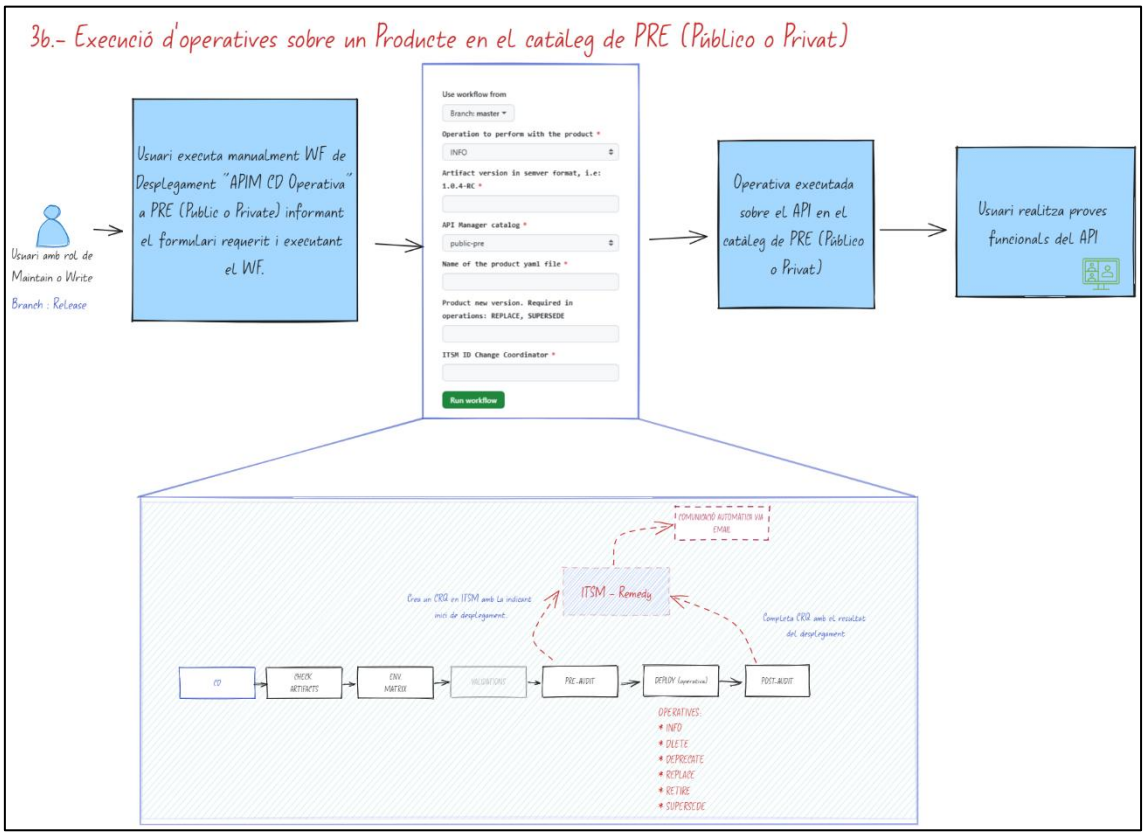


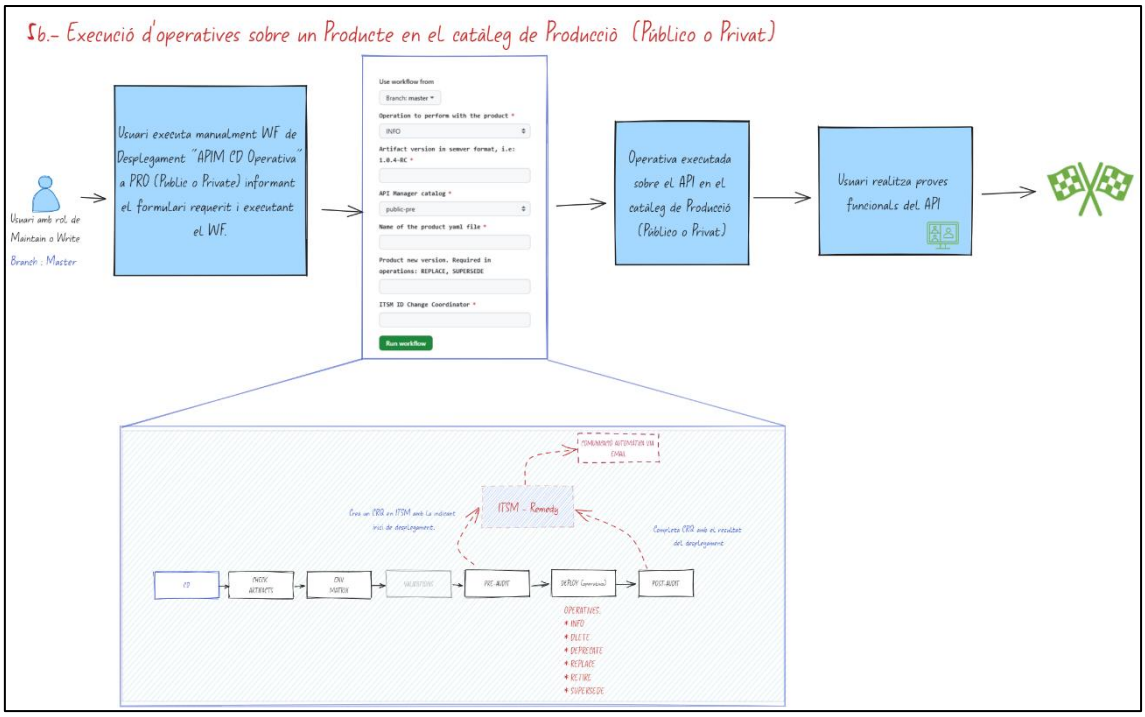
17. Finalment, el desenvolupador ja podrà executar de forma manual el workflow "APIM CD PUBLISH" per desplegar el producte guardat en l'artefacte de GitHub en l'entorn de producció de l'API Manager. Per a això, haurà d'accedir a aquest workflow a la secció d'"Actions", polsar a "Run workflow" i omplir el formulari.

    Els camps necessaris són la branca (el valor de la qual hauria de ser master), versió de l'artefacte (el valor del qual hauria de ser X.X.X), el catàleg on es desplegarà (el valor del qual hauria de ser public o privat), el nom exacte del fitxer YAML del producte i l'ITSM ID Change Coordinator (el valor del qual hauria de ser ID de l’usuari per crear la CRQ en ITSM amb l’objectiu d’informar sobre el desplegament.).

18. Una vez el workflow se ejecute correctamente, se podría comprobar que la subida es correcta tal y como se ha mostrado en pasos anteriores.

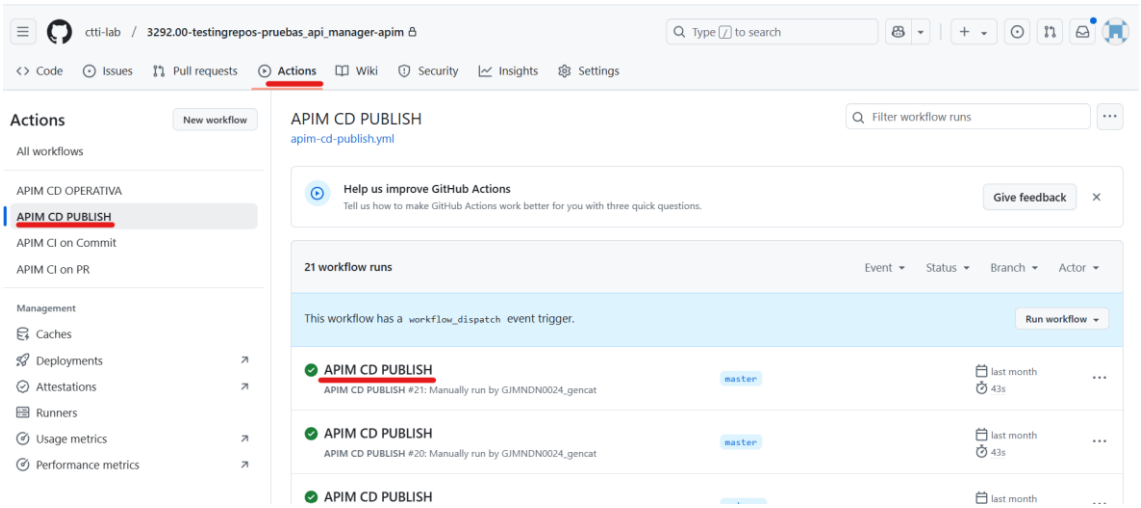
El flux e2e hauria acabat. En cas de voler realitzar alguna de les operatives, una vegada estigui desplegat el producte en aquest entorn, es pot executar de forma manual el workflow "APIM CD OPERATIVA", omplint prèviament el formulari amb les dades adequades. Per a més informació sobre el flux e2e i un exemple complet, accedir a https://canigo.ctti.gencat.cat/plataformes/ghec/workflows/exemples/gh-exemple-e2e-apimanager/



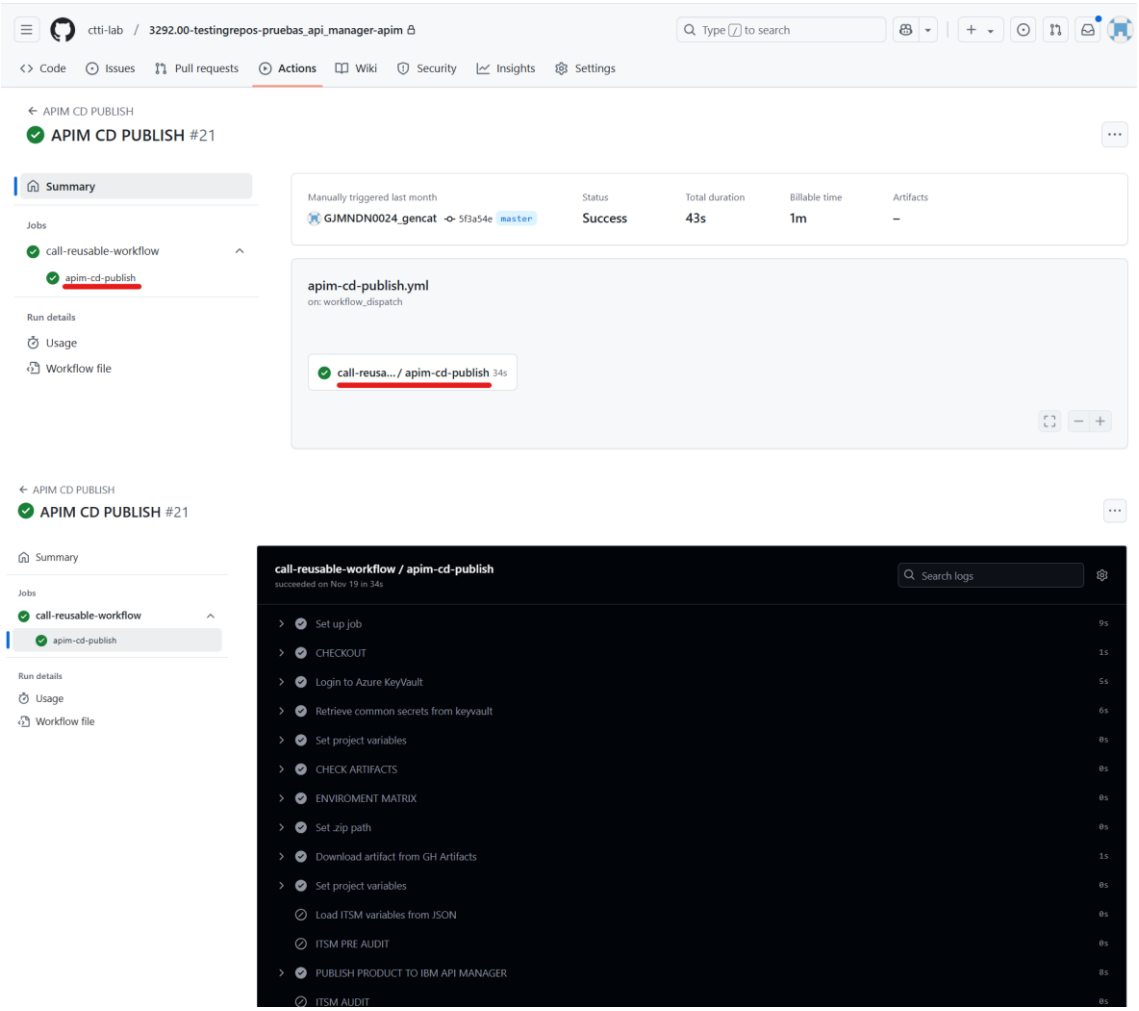


3.2.4.2 Consultar logs

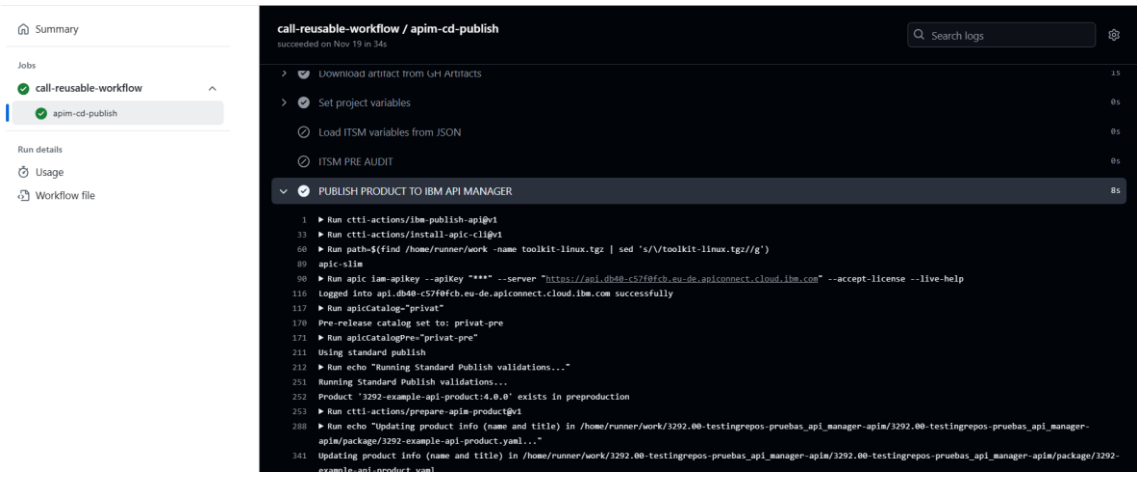
Per consultar els logs de les workflows, accedim a "Actions" i posem sobre l'execució que vulguem revisar.



Posem en qualsevol de les dues zones indicades en la imatge mostrada a continuació.



En aquesta secció es veurà amb detall tots els logs que s' han generat al llarg de l' execució, juntament amb la informació dels errors, si escau.



4 Enllaços d' interès

A continuació, es proporciona una llista enllaços a documents d' interès que puguin servir com a suport:

- Document de preguntes freqüents amb relació al desplegament: https://canigo.ctti.gencat.cat/plataformes/sic/faq/

- Canals de suport de CTTI per comunicar qualsevol incidència: Canals de suport (gencat.cat)

- Funcionament dels pipelins: Autoservei de pipelines (gencat.cat)

- Documentació SIC 3.0: https://canigo.ctti.gencat.cat/plataformes/sic/

- Documentació nou model de CI/CD a cloud públic de SIC+: https://canigo.ctti.gencat.cat/plataformes/ghec/

- Documentació API Manager : https://canigo.ctti.gencat.cat/plataformes/apim/documentacio/